



Gowin FPGA Products

# **Programming and Configuration Guide**

UG290-2.7.2E, 11/16/2023

**Copyright © 2023 Guangdong Gowin Semiconductor Corporation. All Rights Reserved.**

**GOWIN**, Gowin, LittleBee, and GOWINSEMI are trademarks of Guangdong Gowin Semiconductor Corporation and are registered in China, the U.S. Patent and Trademark Office, and other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders. No part of this document may be reproduced or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written consent of GOWINSEMI.

#### **Disclaimer**

GOWINSEMI assumes no liability and provides no warranty (either expressed or implied) and is not responsible for any damage incurred to your hardware, software, data, or property resulting from usage of the materials or intellectual property except as outlined in the GOWINSEMI Terms and Conditions of Sale. All information in this document should be treated as preliminary. GOWINSEMI may make changes to this document at any time without prior notice. Anyone relying on this documentation should contact GOWINSEMI for the current documentation and errata.

## Revision History

Date	Version	Description
4/17/2017	1.00E	Initial version published.
5/31/2017	1.01E	<ul style="list-style-type: none"> <li>● Configuration mode and value of different supported device updated.</li> <li>● RECONFIG_N notes during programming built-in Flash updated.</li> </ul>
10/13/2017	1.02E	Description of reusing pins updated.
3/16/2018	1.03E	GW1NS programming and configuration description added.
8/8/2018	1.04E	<ul style="list-style-type: none"> <li>● The description of configuration process when Flash is empty updated.</li> <li>● Operation procedures for multiple configurations updated.</li> <li>● When MODE[0]=1, JTAG pins reuse description updated.</li> <li>● The programming features of B version devices updated.</li> <li>● Configuration notes and the timing for different configuration modes added.</li> </ul>
1/8/2019	1.05E	<ul style="list-style-type: none"> <li>● The configuration timing and parameters for SERIAL mode added.</li> <li>● The description of power supply requirements deleted.</li> </ul>
8/16/2019	1.06E	<ul style="list-style-type: none"> <li>● Power up description and configuration flow added.</li> <li>● The description of File Size Configuration modified.</li> </ul>
5/15/2020	2.0E	<ul style="list-style-type: none"> <li>● The note of JTAGSEL_N used as IO added.</li> <li>● GW1N(RS/GW1N(R)-2B/GW1N(R)-6 removed.</li> <li>● Configuration mode description optimized.</li> </ul>
8/20/2020	2.1E	<ul style="list-style-type: none"> <li>● JTAG Configuration added.</li> <li>● SSPI Configuration added.</li> <li>● AES Programming added.</li> </ul>
10/30/2020	2.2E	Configuration File Loading Time added.
02/07/2021	2.3E	I <sup>2</sup> C Configuration added.
09/24/2021	2.4E	<ul style="list-style-type: none"> <li>● Configuration process added.</li> <li>● The flow chart of configuring or programming SRAM/Flash for GW1N-2 added.</li> <li>● Process of internal Flash programming added.</li> </ul>
01/20/2022	2.4.1E	Remarks about I <sup>2</sup> C configuration mode added.
05/07/2022	2.5E	<ul style="list-style-type: none"> <li>● Information on GW2AN-9X/18X deleted.</li> <li>● 7.5 MSPI section updated.</li> </ul>
05/10/2022	2.5.1E	CPU Mode Configuration Timing diagram updated.
07/14/2022	2.5.2E	<ul style="list-style-type: none"> <li>● Information on configuration file size added.</li> <li>● Count of address and length of one address of GW1N-2 SRAM updated.</li> <li>● Description of loading frequency for GW1N-2 devices added.</li> </ul>
08/10/2022	2.5.3E	<ul style="list-style-type: none"> <li>● Table 7-6 Gowin FPGA ID CODE updated.</li> <li>● Table 7-9 TCK Frequency Requirements for JTAG updated.</li> </ul>
09/07/2022	2.5.4E	<ul style="list-style-type: none"> <li>● The note on I<sup>2</sup>C configuration mode updated.</li> <li>● The description of READY pin and DONE pin in Table 4-3 Pin Function added.</li> <li>● Table 7-6 Gowin FPGA ID CODE updated.</li> </ul>
10/28/2022	2.6E	<ul style="list-style-type: none"> <li>● Information on GW1NS-2/2C, GW1NSR-2/2C, and GW1NSE-2C removed.</li> <li>● Information on GW1N-1P5 updated.</li> <li>● Description of CLKHOLD_N pin updated.</li> </ul>
11/11/2022	2.6.1E	<ul style="list-style-type: none"> <li>● Section "7.4.5 Connection Diagram for SSPI Configuration</li> </ul>

Date	Version	Description
		<p>Mode" updated.</p> <ul style="list-style-type: none"> <li>● Section "Programming External Flash or Embedded SPI-Flash" updated.</li> <li>● Section "Read Status Register 0x41" updated.</li> <li>● Description of the multiplexing of JTAG pins and JTAGSEL_N pin added.</li> <li>● Figure 3-7 Program AES Key Flow updated.</li> <li>● Process of Erasing T-process FPGAs updated.</li> </ul>
11/24/2022	2.6.2E	<ul style="list-style-type: none"> <li>● Table 7-20 CPU Mode Pins updated.</li> <li>● Table 7-21 CPU Configuration Timing Parameters added.</li> <li>● Figure 7-60 CPU Mode Configuration diagram updated.</li> <li>● Figure 7-61 CPU Mode Configuration Timing added.</li> </ul>
12/02/2022	2.6.3E	<ul style="list-style-type: none"> <li>● Description of the background upgrade feature in section 7.1 Configuration Notes updated.</li> </ul>
01/12/2023	2.6.4E	<ul style="list-style-type: none"> <li>● Note added to Table 7-24 Pin Definition in I2C Configuration Mode.</li> <li>● Table 3-3 Loading Time in MSPI Mode updated.</li> <li>● Table 3-4 Loading Time in Autoboot Mode updated.</li> <li>● Table 10-1 SPI Flash Commands updated.</li> </ul>
01/20/2023	2.6.5E	<ul style="list-style-type: none"> <li>● Note added to Table 10-1 SPI Flash Commands.</li> <li>● Figure 7-60 CPU Mode Configuration diagram updated.</li> <li>● Figure 7-61 CPU Mode Configuration Timing updated.</li> </ul>
02/02/2023	2.6.6E	<ul style="list-style-type: none"> <li>● Table 7-21 CPU Configuration Timing Parameters updated.</li> <li>● Note under Table 10-1 SPI Flash Commands modified.</li> </ul>
02/13/2023	2.6.7E	Note about programming internal Flash using I <sup>2</sup> C added.
06/30/2023	2.7E	<ul style="list-style-type: none"> <li>● Structure of the document adjusted.</li> <li>● Note on RECONFIG_N pin updated.</li> <li>● Table 7-1 Timing Parameters for Power Cycling and RECONFIG_N Triggering(LittleBee® Family) updated.</li> <li>● Table 7-2 Timing Parameters for Power Cycling and RECONFIG_N Triggering(Arora Family).</li> <li>● Screenshots of IDE(based on Gowin_V1.9.9Beta) updated.</li> <li>● Figure 6-1 Configuration Flow updated.</li> <li>● Figure 7-15 Process of Normal Programming updated.</li> <li>● Information on pull-down resistor for MCLK signal added.</li> </ul>
07/24/2023	2.7.1E	<ul style="list-style-type: none"> <li>● Note under Table 10-1 SPI Flash Commands modified.</li> <li>● Description of the JTAGSEL_N pin modified.</li> <li>● Note on the frequency tolerance of the MSPI clock added.</li> </ul>
11/16/2023	2.7.2E	<ul style="list-style-type: none"> <li>● The formula for loading time fixed.</li> <li>● Information on Status Register updated.</li> <li>● Process of Erasing T-process FPGAs updated.</li> <li>● Note under Table 3-1 Gowin FPGA Products Configuration File Size (Max.) modified.</li> <li>● Table 3-2 Maximum Loading Frequency of Configuration File modified.</li> <li>● Note under Table 7-18 Pin Description in MSPI Configuration Mode improved.</li> </ul>

# Contents

<b>Contents .....</b>	<b>i</b>
<b>List of Figures .....</b>	<b>iii</b>
<b>List of Tables .....</b>	<b>vi</b>
<b>1 About This Guide .....</b>	<b>1</b>
1.1 Purpose .....	1
1.2 Related Documents .....	1
1.3 Terminology and Abbreviations .....	2
1.4 Support and Feedback .....	2
<b>2 Glossary .....</b>	<b>3</b>
<b>3 Bitstream File Configuration.....</b>	<b>5</b>
3.1 Configuration Options.....	5
3.2 Configuration Data Encryption (Supported by Arora Family only).....	6
3.2.1 Definition .....	6
3.2.2 Entering the Encryption Key.....	7
3.2.3 Entering the Decryption Key .....	7
3.2.4 Programming Operation .....	8
3.2.5 Programming Flow .....	10
3.3 Configuration File Size .....	13
3.4 Configuration File Loading Time .....	15
<b>4 Configuration Pins.....</b>	<b>18</b>
4.1 Configuration Pin List and Reuse Options .....	18
4.1.1 Configuration Pin List .....	18
4.1.2 Configuration Pin Reuse .....	19
4.2 Configuration Pin Function and Application .....	21
<b>5 Configuration Mode Overview .....</b>	<b>26</b>
5.1 LittleBee® Family of FPGA Products .....	26
5.2 Arora Family of FPGA Products .....	28
<b>6 Configuration Process .....</b>	<b>29</b>
6.1 Power-up Sequence .....	31
6.2 Initialization.....	32

6.3 Configuration .....	32
6.4 Wake-up .....	32
6.5 User Mode .....	33
<b>7 Configuration Mode Details .....</b>	<b>34</b>
7.1 Configuration Notes .....	34
7.2 JTAG Configuration Mode .....	38
7.2.1 JTAG Configuration Mode Pins .....	38
7.2.2 Connection Diagram for the JTAG Configuration Mode .....	40
7.2.3 JTAG Configuration Timing .....	41
7.2.4 JTAG Configuration Process .....	42
7.3 AUTO BOOT Configuration Mode (Supported by LittleBee® Family Only) .....	76
7.4 SSPI Configuration Mode .....	77
7.4.1 SSPI Mode Pins .....	77
7.4.2 SSPI Configuration Timing .....	78
7.4.3 Configuration Instruction .....	78
7.4.4 The Flow Chart of Configuring SRAM via SSPI .....	82
7.4.5 Connection Diagram for SSPI Configuration Mode .....	82
7.4.6 Multiple FPGA Connection View in SSPI Mode .....	88
7.5 MSPI Configuration Mode .....	89
7.5.1 MSPI Mode Pins .....	90
7.5.2 Connection Diagram for MSPI Configuration Mode .....	91
7.5.3 MSPI Mode Configuration Attempts .....	92
7.5.4 MULTI BOOT .....	92
7.5.5 MSPI Configuration Timing .....	96
7.6 DUAL BOOT Configuration Mode (Supported by LittleBee® Family Only) .....	98
7.7 CPU Configuration Mode .....	99
7.7.1 Configuration Timing .....	100
7.8 SERIAL Configuration Mode .....	102
7.9 I <sup>2</sup> C Configuration Mode .....	103
7.9.1 Process of Configuring SRAM of GW1N-2 .....	106
<b>8 Safety Precautions .....</b>	<b>107</b>
<b>9 Boundary Scan .....</b>	<b>109</b>
<b>10 SPI Flash Selection .....</b>	<b>111</b>

# List of Figures

Figure 3-1 Configuration Options .....	6
Figure 3-2 Encryption Key Setting Method .....	7
Figure 3-3 Setting the Decryption Key .....	8
Figure 3-4 AES Security Configuration .....	8
Figure 3-5 Prepare .....	10
Figure 3-6 Read AES Key Flow .....	11
Figure 3-7 Program AES Key Flow .....	12
Figure 3-8 Lock AES Key Flow .....	13
Figure 3-9 Bitstream Format generation .....	14
Figure 4-1 Configuring Pin Reuse .....	21
Figure 4-2 MCLK Frequency Setting .....	24
Figure 6-1 Configuration Flow .....	30
Figure 6-2 POR Power-up Timing .....	31
Figure 7-1 Recommended Pin Connection .....	36
Figure 7-2 Power Cycling Timing .....	37
Figure 7-3 RECONFIG_N Triggering Timing .....	37
Figure 7-4 Connection Diagram for JTAG Configuration Mode .....	40
Figure 7-5 Connection Diagram of JTAG Daisy-Chain Configuration Mode .....	41
Figure 7-6 JTAG Configuration timing .....	41
Figure 7-7 TAP State Machine .....	42
Figure 7-8 Instruction Register Access Timing .....	43
Figure 7-9 Data Register Access Timing .....	43
Figure 7-10 State Machine Flowchart of Reading ID Code .....	45
Figure 7-11 Instruction-0x11 Access Timing When Reading ID Code .....	45
Figure 7-12 Data Register Access Timing When Reading ID Code .....	45
Figure 7-13 SRAM Configuration Flow .....	47
Figure 7-14 Process of reading SRAM .....	49
Figure 7-15 Process of Normal Programming .....	51
Figure 7-16 Process of Background Programming .....	52
Figure 7-17 Process of Erasing T-process FPGAs .....	54
Figure 7-18 Erase Flow for H-process FPGAs .....	56

Figure 7-19 Process of Programming Internal Flash View .....	58
Figure 7-20 X-page Programming .....	59
Figure 7-21 Y-page Programming.....	60
Figure 7-22 Process of Reading Internal Flash .....	61
Figure 7-23 Process of Reading a Y-page.....	62
Figure 7-24 GW1N-4 Background Programming Flow .....	63
Figure 7-25 Transfer JTAG Instruction Sample & Extest Flow Chart .....	64
Figure 7-26 Connection Diagram of Programming External Flash via JTAG Interface (GW2A(R)-18/GW2A-55 /LittleBee® Family) .....	65
Figure 7-27 Connection Diagram of Programming Embedded SPI Flash via JTAG Interface (GW2AN-55) .....	65
Figure 7-28 Process of Programming SPI Flash .....	66
Figure 7-29 Process of Erasing SPI Flash.....	67
Figure 7-30 Process of Programming a Page of the SPI Flash.....	68
Figure 7-31 Process of Reading Back SPI Flash and Verifying the Data Stream File .....	69
Figure 7-32 Timing diagram of Sending 0x06 by Emulating SPI with JTAG(GW2A) .....	70
Figure 7-33 Timing diagram of Sending 0x06 by Emulating SPI with JTAG(GW1N) .....	70
Figure 7-34 Process of Using Boundary Scan Mode To Program SPI Flash .....	71
Figure 7-35 Connection Diagram of Daisy-Chain .....	76
Figure 7-36 SSPI Configuration Timing .....	78
Figure 7-37 Read ID Code Timing .....	79
Figure 7-38 Write Enable (0x15) Timing .....	80
Figure 7-39 Write Disable(0x3A00) Timing .....	80
Figure 7-40 Write Data (0x3B) Timing .....	81
Figure 7-41 SSPI Configuration Mode Connection Diagram.....	82
Figure 7-42 Connection Diagram of Programming External Flash via SSPI(GW2A-18/55,GW1N(R)-9) .....	83
Figure 7-43 Connection Diagram of Programming Internal Flash via SSPI (GW2AN-55) .....	83
Figure 7-44 The Flow Chart of Programming Flash via SSPI .....	84
Figure 7-45 The Flow Chart of Erasing SPI Flash .....	85
Figure 7-46 The Flow Chart of Programming a Page of the SPI Flash .....	86
Figure 7-47 The Flow Chart of Reading Back SPI Flash and Verifying the Data Stream File.....	87
Figure 7-48 Multiple FPGA Connection Diagram 1.....	88
Figure 7-49 Multiple FPGA Connection Diagram 2.....	88
Figure 7-50 Connection Diagram for MSPI Configuration Mode.....	91
Figure 7-51 Connection Diagram of JTAG Programming External Flash.....	91
Figure 7-52 Example of Bitstream Image Distribution in Flash Memory .....	93
Figure 7-53 Input the Start address for the Next Bitstream.....	94
Figure 7-54 Set the Programming Address for the External Flash .....	95
Figure 7-55 Connection Diagram for Configuring Multiple FPGAs via Single Flash.....	96

Figure 7-56 MSPI Download Timing .....	96
Figure 7-57 Multiple FPGA Connection Diagram in MSPI Configuration Mode.....	97
Figure 7-58 Dual Boot Flow Chart .....	98
Figure 7-59 Connection Diagram for CPU Mode.....	100
Figure 7-60 CPU Mode Configuration diagram .....	100
Figure 7-61 CPU Mode Configuration Timing.....	101
Figure 7-62 Connection Diagram for SERIAL Mode.....	102
Figure 7-63 SERIAL Configuration Timing.....	103
Figure 7-64 Connection Diagram for I <sup>2</sup> C Mode .....	104
Figure 7-65 I <sup>2</sup> C Mode Timing .....	104
Figure 7-66 Process of Configuring SRAM of GW1N-2 .....	106
Figure 9-1 Boundary Scan Operation Schematic Diagram .....	109

# List of Tables

Table 1-1 Abbreviations and Terminology .....	2
Table 2-1 Glossary .....	3
Table 3-1 Gowin FPGA Products Configuration File Size (Max.) .....	14
Table 3-2 Maximum Loading Frequency of Configuration File .....	16
Table 3-3 Loading Time in MSPI Mode .....	17
Table 3-4 Loading Time in Autoboot Mode .....	17
Table 4-1 Configuration Pin List .....	18
Table 4-2 Pin Reuse Options .....	20
Table 4-3 Pin Function .....	21
Table 5-1 Configuration Modes .....	27
Table 5-2 Configuration Modes .....	28
Table 6-1 Power Rails Monitored by POR Circuits of Different Devices.....	31
Table 7-1 Timing Parameters for Power Cycling and RECONFIG_N Triggering(LittleBee® Family). 37	
Table 7-2 Timing Parameters for Power Cycling and RECONFIG_N Triggering(Arora Family).....	38
Table 7-3 Pin Description in JTAG Configuration Mode.....	38
Table 7-4 List of devices for which you need/do not need to send a reprogram instruction.....	39
Table 7-5 JTAG Configuration Timing Parameters .....	41
Table 7-6 Gowin FPGA ID CODE .....	44
Table 7-7 Change of TDI and TMS Value in The Process of Sending Instructions .....	44
Table 7-8 Count of Address and Length of One Address .....	48
Table 7-9 TCK Frequency Requirements for JTAG .....	52
Table 7-10 Readback-pattern / Autoboot-pattern.....	57
Table 7-11 Pin State .....	71
Table 7-12 Status Register Definition(I).....	72
Table 7-13 Status Register Definition(II) .....	73
Table 7-14 Status Register Definition(III).....	73
Table 7-15 SSPI Mode Pins .....	77
Table 7-16 SSPI Configuration Timing Parameters .....	78
Table 7-17 Configuration Instruction .....	79
Table 7-18 Pin Description in MSPI Configuration Mode .....	90
Table 7-19 MSPI Configuration Timing Parameters .....	97

Table 7-20 CPU Mode Pins.....	99
Table 7-21 CPU Configuration Timing Parameters.....	101
Table 7-22 Pin Definition in SERIAL Configuration Mode.....	102
Table 7-23 SERIAL Configuration Timing Parameters.....	103
Table 7-24 Pin Definition in I <sup>2</sup> C Configuration Mode.....	104
Table 7-25 I <sup>2</sup> C Configuration Timing Parameters .....	105
Table 7-26 Frequencies and addresses of I <sup>2</sup> C configuration mode.....	106
Table 10-1 SPI Flash Commands .....	111

# 1 About This Guide

## 1.1 Purpose

This guide mainly introduces general features and functions on programming and configuration of LittleBee® family devices and Arora family devices. It helps users to use Gowin FPGA products to their full potential.

## 1.2 Related Documents

The latest user guides are available on the GOWINSEMI Website. You can find the related documents at [www.gowinsemi.com](http://www.gowinsemi.com):

- DS100, [GW1N series of FPGA Products Data Sheet](#)
- DS102, [GW2A series of FPGA Products Data Sheet](#)
- DS117, [GW1NR series of FPGA Products Data Sheet](#)
- DS226, [GW2AR series of FPGA Products Data Sheet](#)
- DS961, [GW2ANR series of FPGA Products Data Sheet](#)
- DS821, [GW1NS series of FPGA Products Data Sheet](#)
- DS841, [GW1NZ series of FPGA Products Data Sheet](#)
- DS861, [GW1NSR series of FPGA Products Data Sheet](#)
- DS871, [GW1NSE series of FPGA Products Data Sheet](#)
- DS881, [GW1NSER series of FPGA Products Data Sheet](#)
- DS891, [GW1NRF series of FPGA Products Data Sheet](#)
- DS961, [GW2ANR series of FPGA Products Data Sheet](#)
- DS976, [GW2AN-55 Data Sheet](#)

## 1.3 Terminology and Abbreviations

The terminology and abbreviations used in this manual are as shown in Table 1-1.

**Table 1-1 Abbreviations and Terminology**

Terminology and Abbreviations	Full Name
Bitstream	Bitstream Data
Bscan	Boundary Scan
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
EFlash/EmbFlash	Embedded Flash
FPGA	Field Programmable Gate Array
FS file	Fuses file
GPIO	General Purpose Input Output
I2C (I <sup>2</sup> C, IIC)	Inter-Integrated Circuits
ID	Identification
IEEE	Institute of Electrical and Electronics Engineers
Internal Flash	Internal Flash
JTAG	Joint Test Action Group
LSB	Least Significant Bit
LUT	Look-up Table
MSB	Most Significant Bit
MSPI	Master Serial Peripheral Interface
Programming	Programming
SCL	Serial Clock
SDA	Serial Data
Security Bit	Security Bit
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SSPI	Slave Serial Peripheral Interface
TAP	Test Access Port

## 1.4 Support and Feedback

Gowin Semiconductor provides customers with comprehensive technical support. If you have any questions, comments, or suggestions, please feel free to contact us directly by the following ways.

Website: [www.gowinsemi.com](http://www.gowinsemi.com)

E-mail: [support@gowinsemi.com](mailto:support@gowinsemi.com)

# 2Glossary

This chapter presents an overview of the terms that are commonly used in the process of programming and configuring Gowin FPGA products to help users get familiar with the related concepts.

**Table 2-1 Glossary**

Glossary	Meaning
Program	Write the bitstream data generated by Gowin Software to the embedded Flash or external SPI Flash of FPGA.
Configure	Load the bitstream data generated by Gowin Software to the FPGA SRAM via external interfaces or embedded Flash.
GowinCONFIG	In addition to the generic JTAG configuration mode, Gowin FPGA products support additional configurations, including AUTO BOOT configuration, DUAL BOOT configuration, MSPI configuration, SSPI configuration, SERIAL configuration, and CPU configuration. How many GowinCONFIG configuration modes each device supports depend on the device model and package.
MODE[2:0]	A representation of the three MODE pin values associated with GowinCONFIG.
AUTO BOOT Configuration	FPGA loads bitstream data into the SRAM from an embedded Flash. Only non-volatile devices support this mode.
DUAL BOOT Configuration	Two bitstream files are stored in embedded Flash and external Flash separately. Switch to the embedded Flash if the external Flash fails to configure. Only non-volatile devices support this mode.
MSPI Configuration	As a master, FPGA is configured by reading bitstream from the external Flash via the SPI interface automatically.
SSPI Configuration	As a slave device, the bitstream data is written into the FPGA via the SPI interface by the external master.
SERIAL Configuration	As a slave device, the bitstream data is written into the FPGA via the serial interface by the external master.

Glossary	Meaning
CPU Configuration	As a slave device, the bitstream data is written into the FPGA via the parallel interface (8-bit) by the external master.
I <sup>2</sup> C Configuration	As a slave device, the bitstream data is written into the FPGA via the I <sup>2</sup> C interface by the external master.
MULTI BOOT	The derivative concept of MSPI, it refers to that FPGA reads bitstream data from different addresses of external Flash. The loading address of the latter bitstream data is written in previous bitstream data and the configuration is completed by triggering RECONFIG_N to switch the data stream file under the condition that the device power is on. FPGA products that support MSPI all support this mode.
Remote Upgrade	After an FPGA starts to work, if an upgrade is required, first write bitstream to an embedded or external Flash through remote operation, and then the FPGA reads the external Flash by triggering RECONFIG_N or powering up again to complete the configuration.
Daisy Chain	FPGA devices are connected sequentially in a serial way. Devices can be configured from the head of the chain in sequence according to the connection order, and data can only be transmitted between adjacent devices.
User Mode	Hands over control to users when the FPGA configuration has been completed. Only in user mode, configuration pins can be reused as GPIOs (Gowin Programmable I/O).
Edit Mode	FPGA can be programmed and configured in this mode. All configuration pins cannot be reused as GPIOs. The output of all GPIOs is high-impedance state, except transparent transmission.
ID CODE	Identification for the Gowin FPGA device. Each series of devices has a different number.
USER CODE	Used to identify the FPGA device that used. The user code can be written to the FPGA device through Gowin programmer. Up to 32-bit can be supported.
Security Bit	A special design for the configuration data security of Gowin FPGA product. After you write the bitstream with security bit to the device SRAM, no one will be able to read back the data. Gowin Software sets a security bit for the bitstream data of all FPGA products by default.
Encryption	Arora family of FPGA products supports this feature. After the encrypted bitstream is written to FPGA, the device will match the pre-stored key automatically, and then decrypt and wake up the device after successful matching. The device cannot work if matching fails.

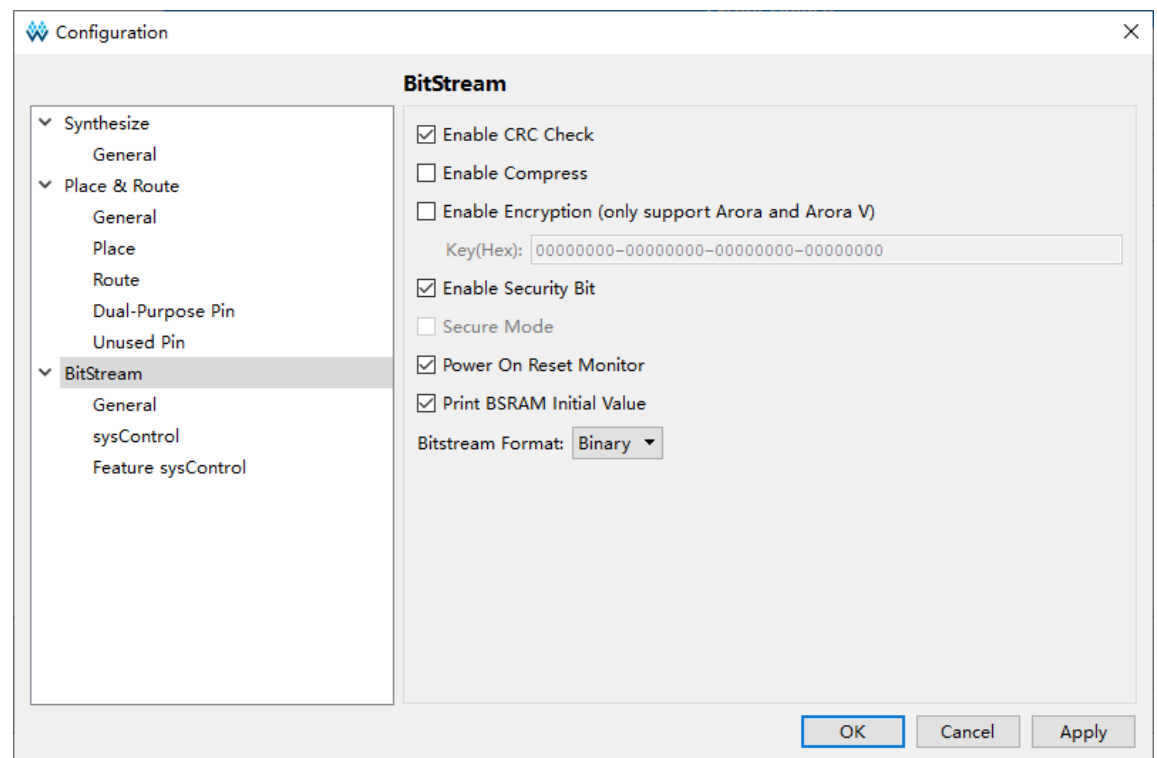
# 3 Bitstream File Configuration

The features of Gowin FPGA products need to be configured and programmed using Gowin Software. The settings mainly include the options of configuration pins multiplexing and bitstream data configuration. This chapter describes the bitstream file configuration. For the details about the configuration pin multiplexing, please refer to [4.1.2 Configuration Pin Reuse](#).

To transfer the configuration data safely and accurately, the CRC calibration algorithm has been incorporated by default in the FPGA bitstream file, and the security bit is set. During the process of data configuration, input data is checked in real time. The wrong data cannot wake up the device, and the DONE signal is pulled down. After the configuration of the bitstream with security bit is complete, data readback cannot be performed.

## 3.1 Configuration Options

Please refer to Figure 3-1 for the related configuration data setting interface. The options include CRC enable, bit stream data compression, encryption key settings, security bit settings, MSPI configuration frequency selection, SPI Flash start address settings in multiple configuration modes, USER CODE setting, etc. The lower 12 bits of an SPI Flash startup address is invalid and the address space of ADDR [23:12] can be set by users.

**Figure 3-1 Configuration Options****Note!**

The security bit setting is forcibly checked after Gowin Software verifies the encryption key setting option. In addition to ensuring the data is secure during the transmission process, using these bitstream settings during configuration also prevents any readback, thereby ensuring maximum protection of user data.

## 3.2 Configuration Data Encryption (Supported by Arora Family only)

The Gowin Arora® Family of FPGA products support bitstream data encryption, using the 128 AES encryption algorithm. Please refer to the following steps for the data encryption configuration:

1. Enter the encryption KEY (KEY) in Gowin Software interface to generate the bitstream data.
2. Enter the decryption key in Gowin Programmer.
3. After encrypted bitstream data is loaded into the device, FPGA compares the data that has been loaded with the decrypt key values stored in advance.

If data parsing succeeds, the device finishes configuration and begins to work; if data parsing fails, the device cannot work, and READY and DONE are pulled down.

### 3.2.1 Definition

- AES encryption key: AES private key used in AES encryption algorithm, specified by users. Referred to as "key" in this manual.
- AES encryption key length: 128 bits.
- Key: An abbreviation for AES encryption key. A 128-bit space is

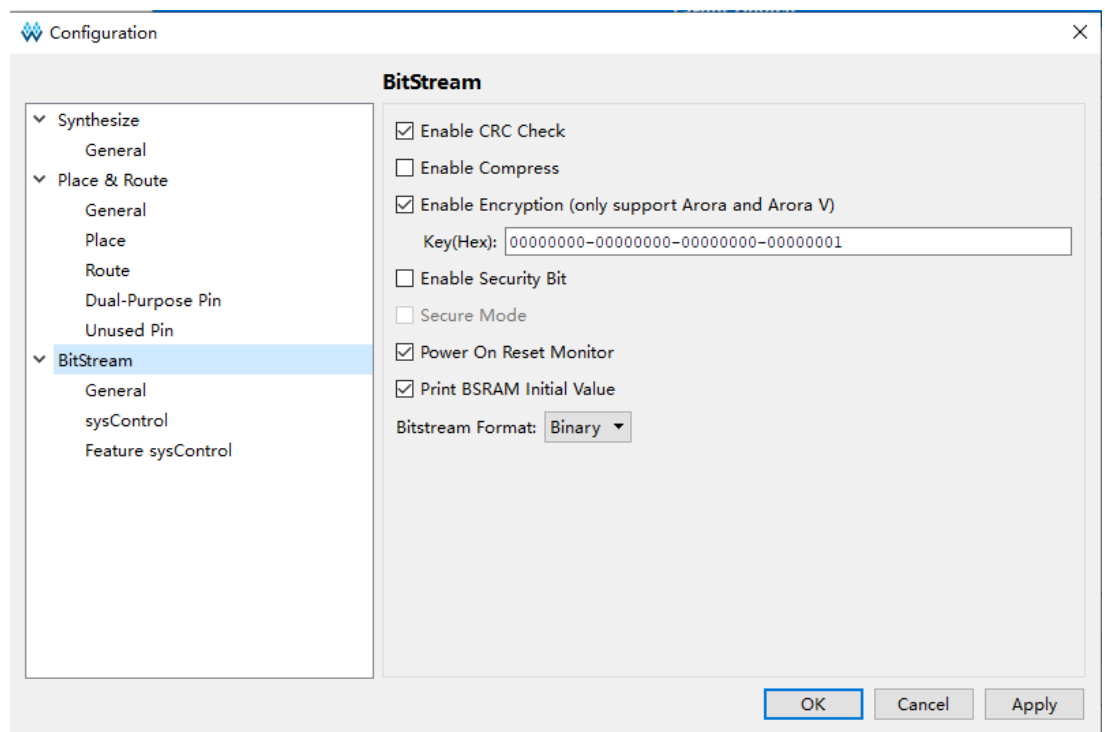
- provided in the GW2A(R) series FPGA products for storing the Key.
- Lock: To ensure the security of AES Key, it is used to control the read permissions for the Key. This operation is named as "lock" in this manual. When it's locked, all the read back data is 1.

### 3.2.2 Entering the Encryption Key

Refer to the steps below to write the encryption keys in Gowin Software:

1. Open the corresponding project in Gowin Software.
2. Select "Project > Configuration > Dual Purpose Pin" from the available menu options.
3. Click "BitStream", check "Enable Encryption (only support GW2A)" and input the key value, as shown in Figure 3-2.

Figure 3-2 Encryption Key Setting Method

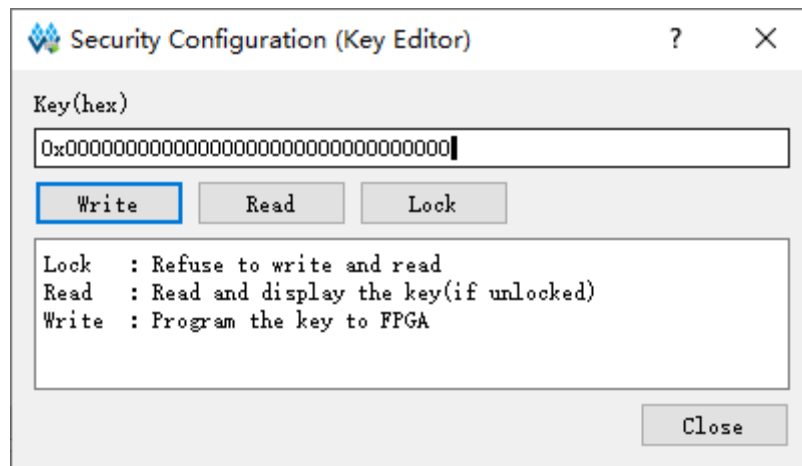


After setting the encryption key successfully, write the decrypted key to the FPGA key storage area for the device to analyze the encrypted bitstream data to complete the configuration.

### 3.2.3 Entering the Decryption Key

To input the decryption key, refer to the following steps:

1. Open the Gowin programming software.
2. Scan the FPGA device.
3. Right-click on the device name and select " Security Key Setting ".
4. Enter the encrypted key value in the pop-up interface, click "write" and write the value to the FPGA, as shown in Figure 3-3.

**Figure 3-3 Setting the Decryption Key**

After the decryption key is written successfully, readback the written value via the "Read" button on the interface to verify.

After the key is written successfully, users also can select to "lock" it in FPGA via the Lock command. Once you have performed this action, any read and write key operations will be invalid, the key value cannot be modified, and all read bits are all "1".

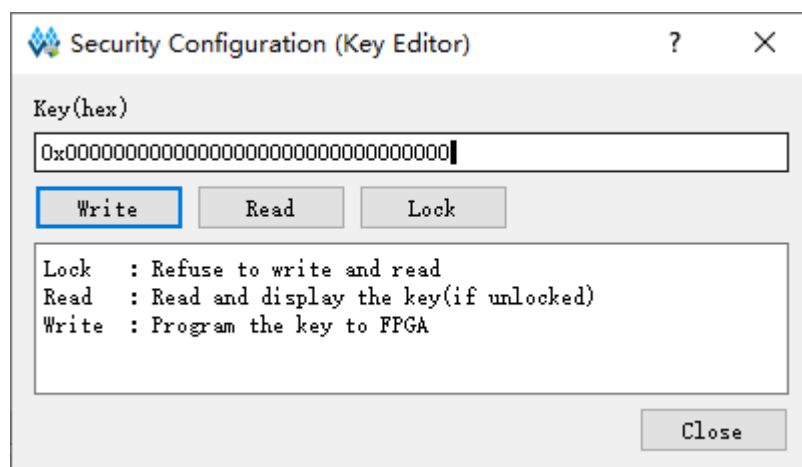
After the decryption key is set, the encrypted bitstream data will only work when the data matches the decryption key. The key does not affect the non-encrypted bitstream data.

**Note!**

The initial value of the Gowin FPGA keys is 0. If a key value is changed to 1, it cannot be changed back to 0. For example, the key value written during an operation is 00000000-00000000-00000000-00000001, and the last bit of the modified key must be 1.

### 3.2.4 Programming Operation

Gowin Programmer offers a tool for programming the AES encryption key. Open this tool by clicking "Edit > Security Key Setting " in Gowin Programmer, as shown in Figure 3-4 .

**Figure 3-4 AES Security Configuration**

This configuration contains the following three parts:

- Write: Write Key.
- Read: Read Key.
- Lock: Lock read and write access to the Key.

### **Write**

1. Write the user-defined Key to the text box in the figure above.
2. Click "write" button.
3. Return the validation result after running.

### **Read**

Click "read" button to validate the written AES encryption key again. The Key that is read from the tool will be displayed in the text box in the figure above.

### **Lock**

Click "lock" to lock the read and write permission of Key. If it is locked, the Key cannot be read or written.

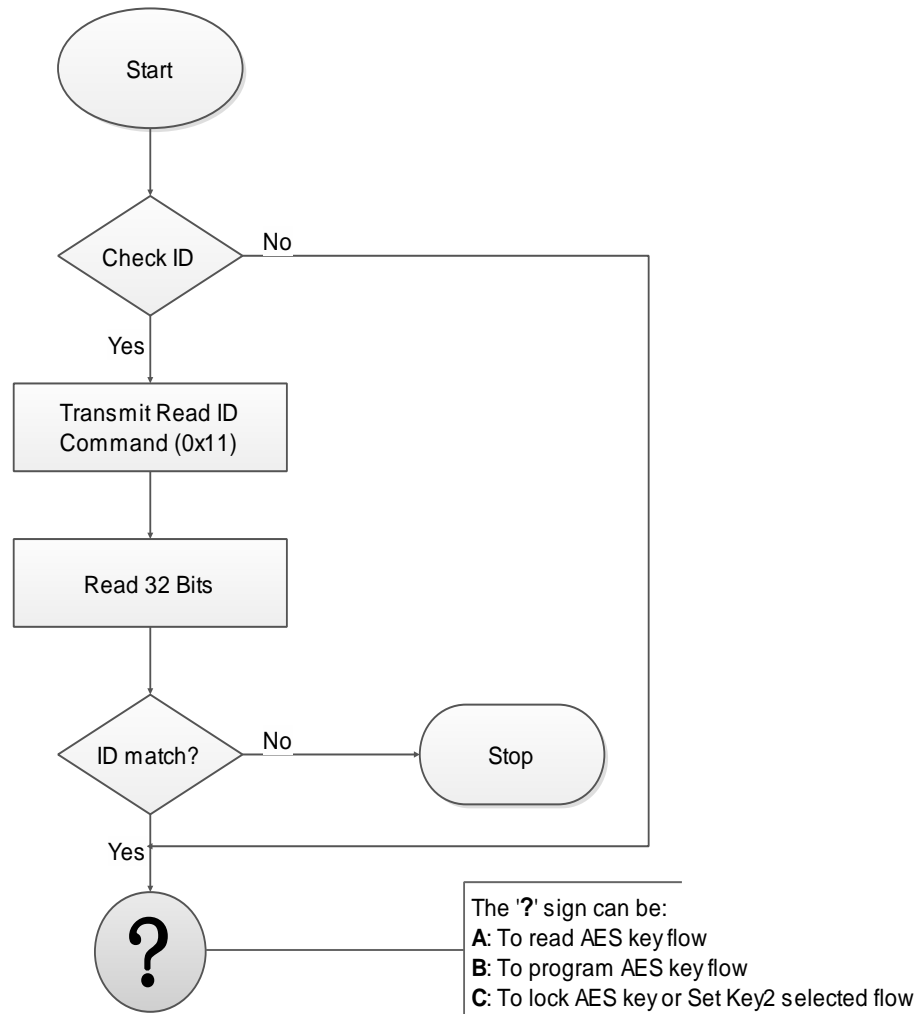
### 3.2.5 Programming Flow

Figure 3-5 ~ Figure 3-8 show the flow of how to program or lock the AES key. All the flows are based on JTAG protocol.

#### Check ID CODE

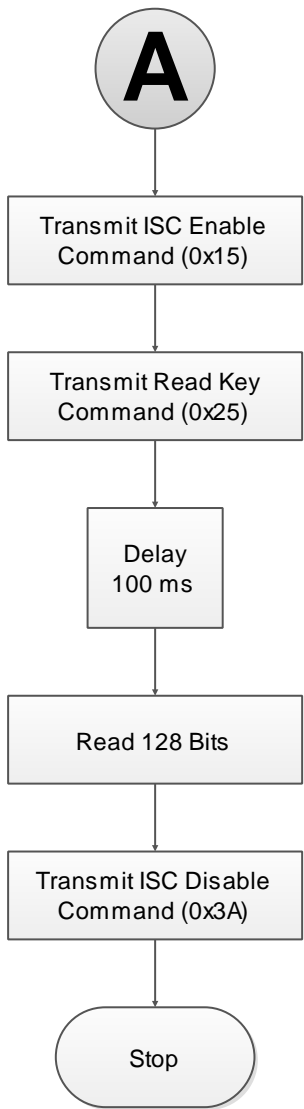
Check the device ID to determine whether the JTAG protocol works properly and whether the programming object is correct to avoid misoperation.

Figure 3-5 Prepare



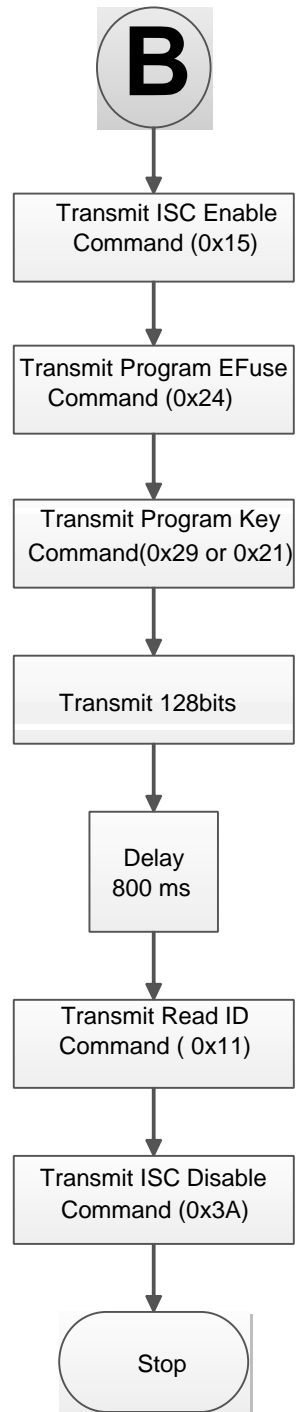
Read AES Key

Figure 3-6 Read AES Key Flow



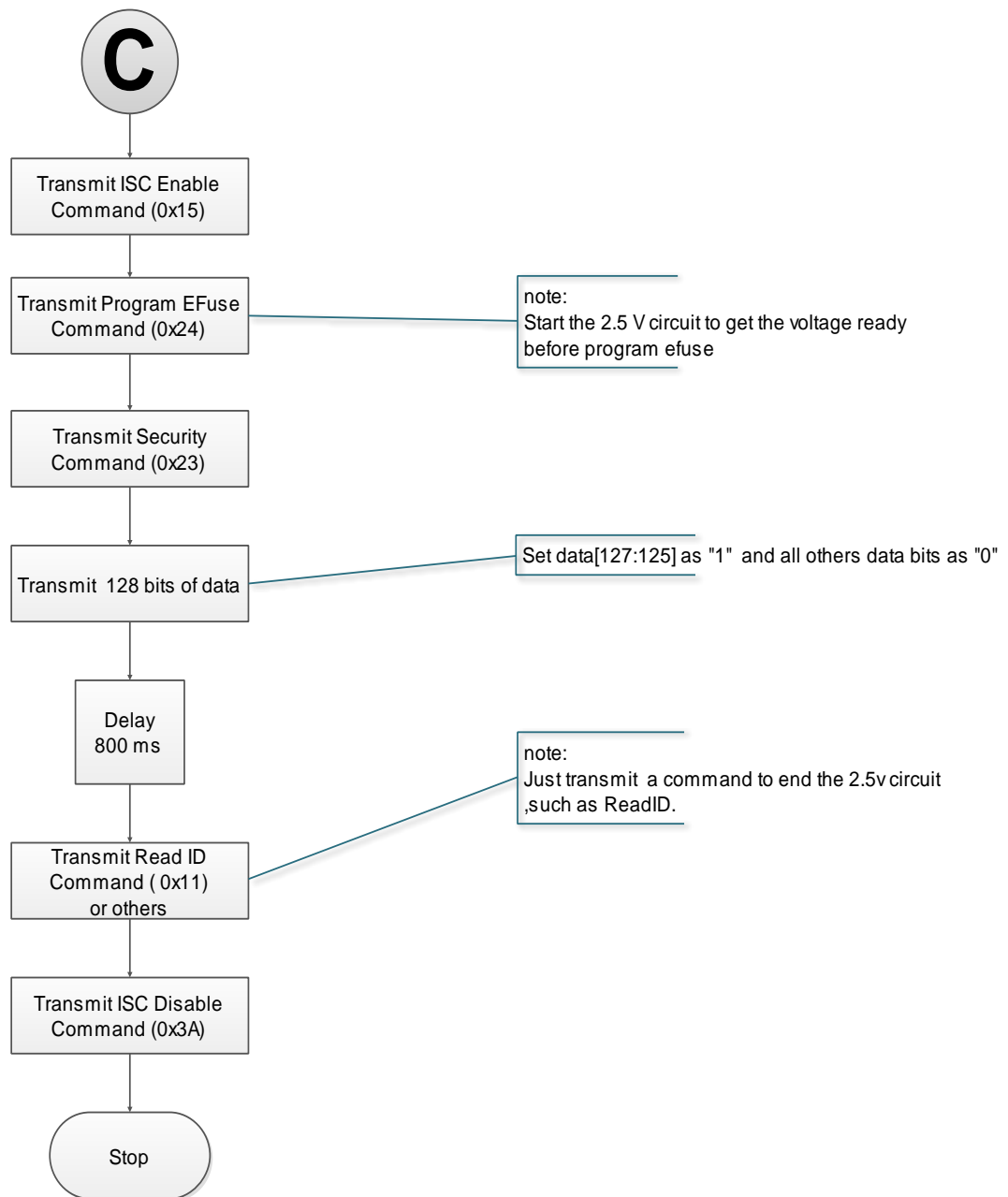
## Program AES Key

Figure 3-7 Program AES Key Flow



## Lock AES Key

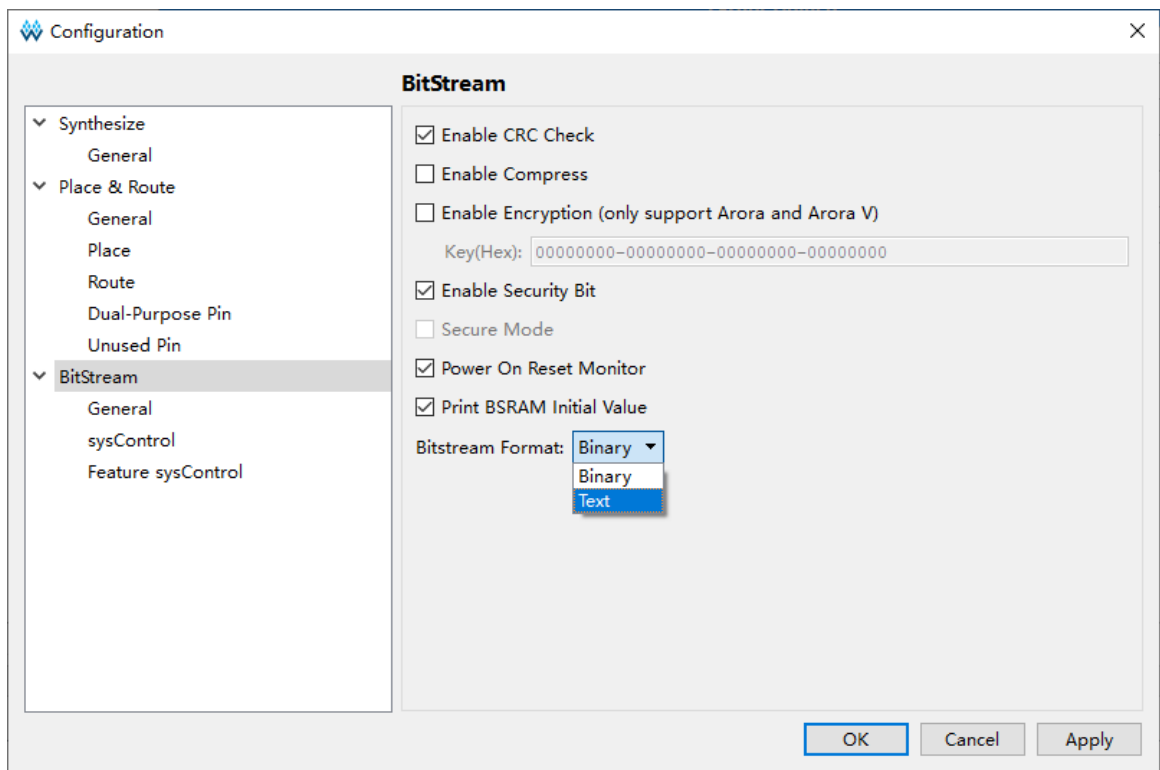
Locking the AES Key prevents the Key leakage. After locking the AES Key, you will not be able to read or configure the AES Key.

**Figure 3-8 Lock AES Key Flow**

## 3.3 Configuration File Size

The Gowin bitstream format can be Text (ASCII) with annotations or Binary with no annotations. The file with a .fs suffix is a text format file. Lines beginning with “//” are annotations. The others is the bitstream data. The file with a .bin suffix is a binary format file, with no annotations. This binary format file is commonly used for embedded programming. Users can configure the bitstream file format in Gowin Software.

1. Open the Gowin Software.
2. On the Process tab, right click Place & Route and then click “Configuration > Bitstream”.
3. In the options of Bitstream Format, select Text or Binary, as shown in Figure 3-9.

**Figure 3-9 Bitstream Format generation**

Gowin supports compressing bitstream data. The compression ratio is related to the user design. This manual only provides uncompressed configuration file sizes, as shown in Table 3-1.

**Table 3-1 Gowin FPGA Products Configuration File Size (Max.)**

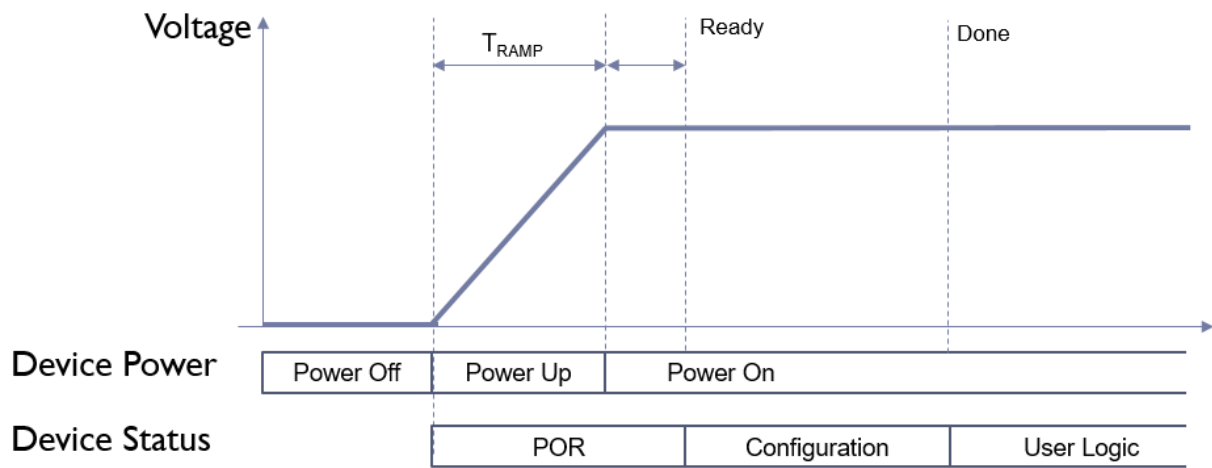
Device Name	LUT	Max. Configuration File Size
GW1N-1(S), GW1NR-1, GW1NZ-1	1,152	84 KBytes
GW1N-1P5	1,584	113 KBytes
GW1N-2, GW1NR-2	2,304	113 KBytes
GW1N-4, GW1NR-4, GW1NS-4(C), GW1NSR-4(C), GW1NSER-4C, GW1NRF-4B	4,608	217 KBytes
GW1N-9, GW1NR-9	8,640	435 KBytes
GW2A-18, GW2AR-18, GW2ANR-18	20,736	887 KBytes
GW2A-55, GW2AN-55	54,720	2269 KBytes

**Note!**

The data in the table is the file size in binary format, and the configuration file is not compressed.

## 3.4 Configuration File Loading Time

Gowin FPGA can be used as Master to read bitstream files from Flash and configure SRAM, including Autboot mode and MSPI mode. In Autboot mode, FPGA reads bitstream files from internal Flash. In MSPI mode, FPGA reads bitstream files from external Flash. When the FPGA is powered on and ready, it starts to read bitstream files, and when the loading is done, the FPGA enters the User Logic state, as shown in the figure below.



Both LittleBee<sup>®</sup> family and Arora family of GOWINSEMI FPGA devices support the MSPI mode, that is, after the device is powered on, it can read bitstream files from the external SPI Flash and then complete the configuration. The default frequency of reading configuration file is 2.5 MHz. One bit is read at each SPI clock, so the required loading time can be calculated according to the file size. The clock frequency of reading SPI Flash in MSPI mode should not be greater than 66.6MHz. Note that the FastRead\_n pin should be grounded at the same time when Fast Read SPI (0x0B) is used.

The LittleBee<sup>®</sup> family devices support not only MSPI mode, but also Autboot mode. The loading frequency is 2.5 MHz by default, and Autboot mode loads one byte (8 bits) per clock.

**Note!**

- For the GW1N-2 device, if its MODE[2] value is fixed to 1, its loading frequency can only be 2.5MHz.

The loading time varies depending on the configuration file size, load frequency, and per-clock loading width. Due to the different process of the embedded Flash, the maximum Autboot loading frequency for different devices is also different. The specific maximum loading speed is as shown in Table 3-2 below.

**Table 3-2 Maximum Loading Frequency of Configuration File**

Device	Max. Loading Frequency in Autoboot mode
GW2A-55/55C	-
GW2A-18/18C	
GW2AR-18/18C	
GW2ANR-18C	
GW1N-1	26MHz
GW1N-1S	
GW1NZ-1 GW1N-2/1P5 GW1N-2B/1P5B GW1NSER-4C GW1NS-4 GW1NSR-4 GW1NS-4C GW1NSR-4C GW1N-4B GW1NR-4B GW1NRF-4B GW1N-4 GW1NR-4 GW1N-9 GW1N-9C GW1NR-9 GW1NR-9C	40MHz

The bitstream file loading time in MSPI mode is as shown in Table 3-3.

**Table 3-3 Loading Time in MSPI Mode**

Number of LUT4	Max. Configuration File	Loading Time (ms, when Frequency =2.5 MHz)	Loading Time (ms, when Frequency =25 MHz)	Loading Time (ms, when Frequency =41.6 MHz)	Loading Time (ms, when Frequency =62.5 MHz)
1,152	84 KBytes	275	28	17	11
1584	116 KBytes	381	40	25	17
2304	116 KBytes	381	40	25	17
4,608	217 KBytes	711	71	42	28
8,640	435 KBytes	1425	142	85	57
20,736	887 KBytes	2906	290	174	116
54,720	2269 KBytes	7435	743	446	297

The bitstream file loading time in Autoboot mode is as shown in Table 3-4.

**Table 3-4 Loading Time in Autoboot Mode**

Number of LUT4	Max. Configuration File	Loading Time (ms, when frequency =2.5 MHz, default frequency)	Loading Time (ms, when Frequency =25 MHz)	Loading Time (ms, when Frequency =31.25 MHz)
1,152	84 KBytes	34	4	3
1584	116 KBytes	48	7	6
2304	116 KBytes	48	7	6
4,608	217 KBytes	88	9	7
8,640	435 KBytes	178	17	14

The loading time listed above is for reference only. The total time required from the device power-up to the completion of configuration includes the power supply ramp time (Tramp), device initialization time, and configuration time. The power supply ramp time is dependent on the characteristics of the power supply device. Therefore, the time from power-up to the completion of loading of the FPGA can be estimated using the following formula:

**Autoboot mode:**

$T_{\text{loading time}} = \text{POR time} + \text{Number of Data Stream Bits} / 8 / \text{Loading Frequency}$

**MSPI mode:**

$T_{\text{loading time}} = \text{POR time} + \text{Number of Data Stream Bits} / \text{Loading Frequency}$

# 4 Configuration Pins

Gowin FPGA products have various configuration modes, including general JTAG configuration, active configuration, passive configuration, serial configuration and parallel configuration, etc., which can meet the various peripheral requirements of different users. The programming and configuration pins can be used as configuration pins and also can be reused as GPIO. Users can configure the pins as required. Users also can configure them according to their configuration functions to meet specific requirements.

## 4.1 Configuration Pin List and Reuse Options

### 4.1.1 Configuration Pin List

Table 4-1 contains a list of all the configuration pins of Gowin FPGA products together with the details of the pins used in each configuration mode and the shared pins in chip packages.

**Table 4-1 Configuration Pin List**

Pin Name	I/O	JTAG	GowinCONFIG						
			AUTO BOOT	I <sup>2</sup> C	SSPI	MSPI	DUAL BOOT	SERIAL	CPU
RECONFIG_N	I	√	√	√	√	√	√	√	√
JTAGSEL_N	I	√							
TDO	O	√							
TMS	I	√							
TCK	I	√							
TDI	I	√							
READY	I/O	√	√	√	√	√	√	√	√
DONE	I/O	√	√	√	√	√	√	√	√
MODE[2:0]	I		√	√	√	√	√	√	√
SCLK	I				√			√	√

Pin Name	I/O	JTAG	GowinCONFIG						
			AUTO BOOT	I <sup>2</sup> C	SSPI	MSPI	DUAL BOOT	SERIAL	CPU
CLKHOLD_N/DIN	I				√			√	√
WE_N/DOUT	O							√	√
MI /D7	I/O					√			√
MO /D6	I/O					√			√
MCS_N /D5	I/O					√			√
MCLK /D4	I/O					√			√
FASTRD_N /D3	I/O					√			√
SI /D2	I/O				√				√
SO /D1	I/O				√				√
SSPI_CS_N/D0	I/O				√				√
SCL	I			√					
SDA	I/O			√					

**Note!**

- For the configuration modes supported by different devices, please refer to [5 Configuration Mode](#).
- Please refer to [7 Configuration Mode](#) for the definition of each pin in different configuration modes.

## 4.1.2 Configuration Pin Reuse

To maximize the utilization of I/O, Gowin FPGA products support setting the configuration pins as GPIO pins. Before any configuration operation is performed on all series of Gowin FPGA products after power up, all related configuration pins are used as configuration pins by default. After successful configuration, the device enters into user mode and reassigns the pin functions according to the multiplex options selected by the user.

**Note!**

When setting the pin multiplexing option, ensure the external initial connection state of the pins does not affect the device configuration. Isolate the connections that affect the configuration first, and then wait to modify them in user mode.

The reuse options for the configuration pins are detailed in Table 4-2.

**Table 4-2 Pin Reuse Options**

Name	Options	Description
JTAG PORT	Default Status	TMS, TCK, TDI, and TDO are used as dedicated configuration pins. JTAGSEL_N is used as GPIO.
	Set as GPIO	JTAGSEL_N pins are used as dedicated configuration pins: <ul style="list-style-type: none"> <li>● JTAGSEL_N=0, TMS, TCK, TDI, and TDO are used as configuration pins:</li> <li>● JTAGSEL_N = 1, TMS, TCK, TDI, and TDO are used as GPIO after configuration.</li> </ul>
I <sup>2</sup> C PORT	Default Status	SCL and SDA pins are used as dedicated configuration pins.
	Set as GPIO	SCL and SDA pins are used as GPIO after configuration.
SSPI PORT	Default Status	SCLK, CLKHOLD_N, SSPI_CS_N, SI and SO are used as dedicated configuration pins.
	Set as GPIO	SCLK, CLKHOLD_N, SSPI_CS_N, SI and SO are used as GPIO after configuration.
MSPI PORT	Default Status	FASTRD_N, MCLK, MCS_N, MO and MI are used as dedicated configuration pins.
	Set as GPIO	FASTRD_N, MCLK, MCS_N, MO and MI are used as GPIO after configuration.
RECONFIG_N	Default Status	Dedicated configuration pins.
	Set as GPIO	Used as GPIO after configuration.
READY	Default Status	Dedicated configuration pins.
	Set as GPIO	Used as GPIO after configuration.
DONE	Default Status	Dedicated configuration pins.
	Set as GPIO	Used as GPIO after configuration.

**Note!**

- [1] For the devices with JTAGSEL\_N unbound, when debugging JTAG pin reuse, it's suggested to set MODE value to non-auto configuration mode (being neither auto-boot, dual boot, nor MSPI) before power up to avoid the other bit stream data affecting configuration. Device turns into user MODE, and JTAG pin changes into GPIO after power up and manually configuring JTAG. After the device is power up, the device enters User Mode, and the JTAG pin is used as GPIO. For the LittleBee® Family of FPGA products, when MODE[2: 0]=001, the JTAGSEL\_N pin is always a GPIO, in other words the JTAGSEL\_N pin and the four JTAG pins (TCK, TMS, TDI, TDO) can be used as GPIOs simultaneously; however, in this case the JTAGSEL\_N pin cannot restore the JTAG pins to configuration IOs, and these pins will be restored as configuration IOs when the FPGA re-enters edit mode.
- [2] The pins of SERIAL and CPU modes are shared with other configuration modes, so they cannot be set as GPIOs separately. However, the pins can be set as GPIOs in non-shared configuration modes.

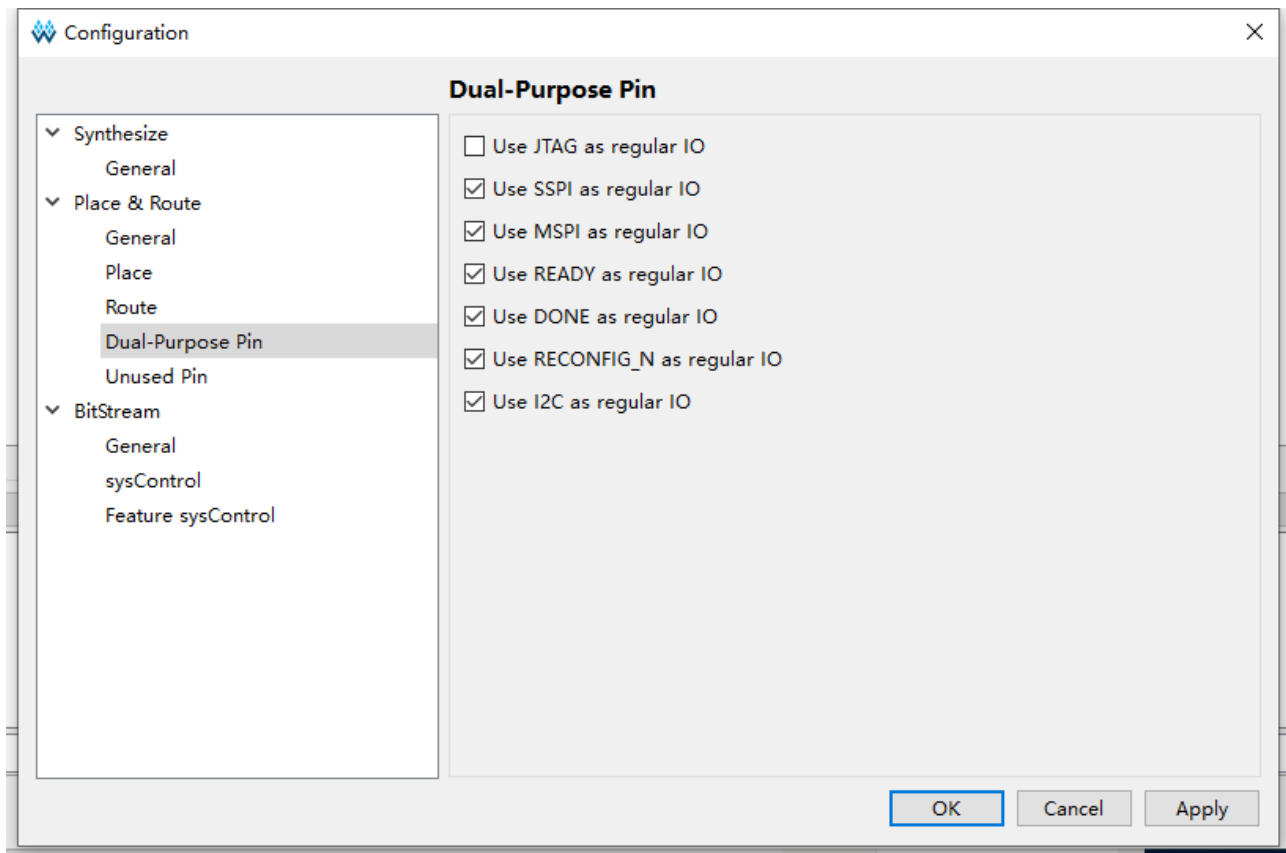
**Configuring Dual-purpose Pins**

The steps are as follows:

1. Open the project in Gowin Software.
2. Select "Project > Configuration > Dual Purpose Pin" from the menu options, as shown in Figure 4-1.

3. Check the corresponding options.

Figure 4-1 Configuring Pin Reuse



## 4.2 Configuration Pin Function and Application

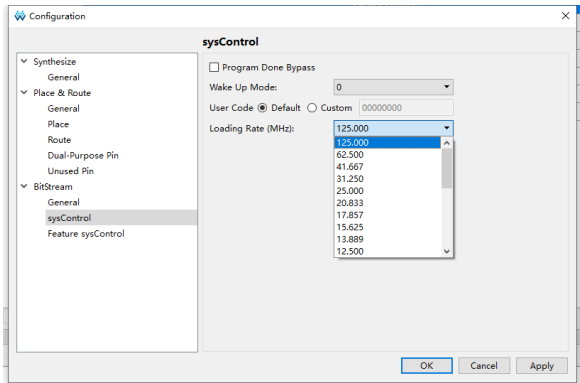
The RECONFIG\_N, READY, and DONE pins are used in all configuration modes. Other pins can be set as dedicated pins or GPIO (Gowin Programmable IO) according to their specific application.

Table 4-3 Pin Function

Pin Name	Functional Description
RECONFIG_N	As a configuration pin, RECONFIG_N is an input pin that has an internal weak pull-up. Active low. It is used as the reset function for the FPGA programming configuration. The <b>FPGA can't be configured if RECONFIG_N is set to low. It is important to keep this pin high or floating during power-up, initialization, and configuration of the FPGA, and it can be released after configuration is complete.</b> As a configuration pin, a low level signal with pulse width no less than 25ns is required for GowinCONFIG to reload bitstream data according to the MODE setting value. You can also write logic to control the pin to trigger the device to reconfigure as required. <b>As a GPIO pin, RECONFIG_N can only be used as the output pin.</b> To ensure a smooth configuration, you need to set the initial value of RECONFIG_N to high when reusing it.
READY	In-out pins. The default state is open-drain output with internal weak pull-up. Active-high. FPGA can be configured only when the

Pin Name	Functional Description
	<p>READY signal is pulled up. When the READY signal is pulled down, recover the status by powering up or triggering RECONFIG_N.</p> <p>As an output configuration pin, it indicates that the FPGA can be configured or not. If the FPGA meets the configuration condition, the READY signal is high. If the configuration fails, READY signal is low. As an input configuration pin, you can delay the configuration via its own logic or by pulling down the READY signal.</p> <p>As a GPIO, it can be used as an input or output type. <b>If READY is used as an input GPIO, the initial value needs to be 1 before configuration. Otherwise, the FPGA cannot be configured.</b></p>
<b>DONE</b>	<p>In-out pins. The default state is open-drain output with internal weak pull-up, and DONE output 0 during configuration. A signal which indicates FPGA is configured successfully, DONE is pulled up after successfully configuring.</p> <p>As an output configuration pin, it indicates the current configuration of FPGA: if configured successfully, the DONE signal is high and the device enters into working state. if the configuration fails, the DONE signal keeps low. As an input configuration pin, the user can delay the entering of user mode via its own internal logic or by reducing the DONE signal. When RECONFIG_N or READY signals are low, DONE signal also keeps low. When configuring SRAM using JTAG circuit, it does not need to take DONE signal into account.</p> <p>As a GPIO, it can be used as an input or output type. <b>If DONE is used as an input GPIO, the initial value of DONE should be 1 before configuring. Otherwise, the FPGA will fail to enter the user mode after being configured.</b></p>
<b>MODE</b>	<p>GowinCONFIG modes selection pin. As the selection pin of GowinCONFIG modes, MODE is an input pin that has internal weak pull-up. The maximum bit width is 3 bits. When FPGA powers up or a low level pulse triggers RECONFIG_N, the device enters the corresponding GowinCONFIG mode in accordance with the MODE value. The same MODE value of the different Gowin series of FPGA products may have different configuration MODE. As the number of pins for each package is different, some MODE pins are not all bonded out, please refer to the corresponding PINOUT manual for further details.</p> <p>When MODE pins are used as GPIOs, they can be used as an input or output type.</p> <p>Note that when the MODE value changes, power cycling or providing one low pulse for triggering RECONFIG_N is required for it to take effect.</p>
<b>JTAGSEL_N</b>	<p>As a configuration pin, it is an input pin with internal weak pull-up. If JTAG pins are set as a GPIO in the Gowin Software, the JTAG pins can become GPIOs after the device being powered up and successfully configured. The JTAG pin configuration functions can be recovered by pulling down JTAGSEL_N. The JTAG</p>

Pin Name	Functional Description
	<p>configuration functions are always available if no JTAG pin reuse is set. As a GPIO, it can be used as an input or output type.</p> <p><b>Note!</b></p> <p>The JTAGSEL_N pin and four JTAG pins (TCK, TMS, TDI, and TDO) are exclusive. JTAG pins can only be used as configuration pins if JTAGSEL_N is set as a GPIO. JTAGSEL_N can only be used as a configuration pin if JTAG pins are set as GPIOs.</p> <p>For the LittleBee® Family of FPGA products, when MODE[2:0]=001, the JTAGSEL_N pin is always a GPIO, in other words the JTAGSEL_N pin and the four JTAG pins (TCK, TMS, TDI, TDO) can be used as GPIOs simultaneously; however, in this case the JTAGSEL_N pin cannot restore the JTAG pins to configuration IOs, and these pins will be restored as configuration IOs when the FPGA re-enters edit mode.</p>
<b>TCK</b>	<p>As a configuration pin, it is an input pin.</p> <p>It is a serial clock input pin in the JTAG configuration mode. As a GPIO, it can be used as an input or output type.</p>
<b>TMS</b>	<p>As a configuration pin, it is an input pin with internal weak pull-up.</p> <p>It is a serial input pin in JTAG configuration mode. As a GPIO, it can be used as an input or output type.</p>
<b>TDI</b>	<p>As a configuration pin, it is an input pin with internal weak pull-up.</p> <p>It is a serial data input pin in JTAG configuration mode. As a GPIO, it can be used as an input or output type.</p>
<b>TDO</b>	<p>As a configuration pin, it is an output pin.</p> <p>It is a serial data output pin in JTAG configuration mode. As a GPIO, it can be used as an input or output type.</p>
<b>SCLK</b>	<p>As a configuration pin, it is an input pin.</p> <p>It is a clock input pin in SSPI, SERIAL, and CPU configuration modes. As a GPIO, it can be used as an input or output type.</p>
<b>CLKHOLD_N</b>	<p>As a configuration pin, it is an input pin with internal weak pull-up.</p> <p>It is a clock-locking pin in SSPI and CPU configuration modes: active high in SSPI mode; active low in CPU mode. As a GPIO, it can be used as an input or output type.</p>
<b>SSPI_CS_N</b>	<p>As a configuration pin, it is an input pin with internal weak pull-up.</p> <p>It is a chip selection signal in the SSPI configuration mode, active low. As a GPIO, it can be used as an input or output type.</p>
<b>SI</b>	<p>As a configuration pin, it is an input pin. It is a serial data input pin in the SSPI configuration mode. As a GPIO, it can be used as an input or output type.</p>
<b>SO</b>	<p>As a configuration pin, it is an output pin. It is a serial data output pin in the SSPI configuration mode. As a GPIO, it can be used as an input or output type.</p>

Pin Name	Functional Description
<b>MCLK</b>	<p>As a configuration pin, it is an output pin.</p> <p>The output clock pin in the MSPI configuration mode is generated from a crystal oscillator in FPGA. The output frequency range of the crystal oscillator is 2.5 MHz ~ 125 MHz, and the default output frequency is 2.5 MHz. The MSPI configuration mode does not support 125 MHz clock. Please refer to the corresponding device datasheet for further detailed data on the on-chip crystal oscillator. The MCLK frequency values can be modified through the Gowin Software interface<sup>[1]</sup>, as shown in Figure 4-2. Open Gowin Software, select "Project &gt; Configuration" from the menu options, click "BitStream" and select the MCLK frequency values from the "sysControl" pull-down list. As a GPIO, it can be used as an input or output type.</p> <p>Note!</p> <p>The MSPI configuration mode clock frequency has a tolerance of <math>\pm 10\%</math>(Arora family) or <math>\pm 5\%</math>(LittleBee family).</p> <p><b>Figure 4-2 MCLK Frequency Setting</b></p> 
<b>MCS_N</b>	<p>As a configuration pin, it is an output pin.</p> <p>It is a chip selection signal in MSPI configuration mode, active low. As a GPIO, it can be used as an input or output type.</p>
<b>MI</b>	<p>As a configuration pin, it is an input pin.</p> <p>It is a serial data input pin in MSPI configuration mode. As a GPIO, it can be used as an input or output type.</p>
<b>MO</b>	<p>As a configuration pin, it is an output pin.</p> <p>Serial data output pin in MSPI configuration mode. As a GPIO, it can be used as an input or output type.</p>
<b>FASTRD_N</b>	<p>As a configuration pin, it is an input pin.</p> <p>In the MSPI mode, FASTRD_N is used to select Flash access speed. High indicates regular Flash access mode(command 0x03). Low indicates high-speed Flash access mode;</p> <p>The high-speed flash access command of each manufacturer is different. Please refer to the corresponding Flash manual. As a GPIO, it can be used as an input or output type.</p>
<b>WE_N</b>	<p>As a configuration pin, it is an input pin.</p> <p>Select the data input/output of D[7:0] in CPU mode: Read operation when WE_N is high; write operation when WE_N is low. As a GPIO, it can be used as an input or output type.</p>

Pin Name	Functional Description
<b>D0 ~ D7</b>	In-out pins. Data input/output pins in CPU configuration mode, 8-bit width. Determine the input/output of D0 ~ D7 according to WE_N. As a GPIO, it can be used as an input or output type.
<b>DIN</b>	As a configuration pin, it is an input pin with internal weak pull-up. It is a serial data input pin in the SERIAL configuration mode. As a GPIO, it can be used as an input or output type.
<b>DOUT</b>	As a configuration pin, it is an output pin. It is a serial data output pin in the SERIAL configuration mode, which is only used as the input to the latter device when the FPGA is cascading. As a GPIO, it can be used as an input or output type.
<b>SCL</b>	As a configuration pin, it is an input pin. As a GPIO, it can be used as an input type.
<b>SDA</b>	As a configuration pin, it is an in/out pin. As a GPIO, it can be used as an input or output type.

# 5 Configuration Mode Overview

## 5.1 LittleBee® Family of FPGA Products

Besides the JTAG configuration mode that is commonly used in the industry, the LittleBee® Family of FPGA products also support GOWINSEMI's own configuration mode: GowinCONFIG. GowinCONFIG configuration modes that are available and supported for each device depend on the device model and package. All non-volatile devices support JTAG and AUTO BOOT modes. Up to six configuration modes are supported, as shown in Table 5-1.

**Table 5-1 Configuration Modes**

Configuration Modes		MODE[2:0] <sup>[1]</sup>	Description
JTAG		XXX <sup>[2]</sup>	The LittleBee® Family of FPGA products are configured via JTAG interface by external Host.
GowinCONFIG	AUTO BOOT	000	FPGA reads data from embedded Flash for configuration
	I <sup>2</sup> C <sup>[6]</sup>	100	FPGA products are configured via I <sup>2</sup> C interface by external Host.
	SSPI	001	FPGA products of LittleBee® Family are configured via SPI interface.
	MSPI	010	As a Master, FPGA reads data from an external Flash (or other devices) via the SPI interface <sup>[3]</sup> .
	DUAL BOOT <sup>[4]</sup>	110	FPGA reads data from external Flash first and if the external Flash configuration fails, it reads from the Internal Flash.
	SERIAL <sup>[5]</sup>	101	External Host configure FPGA products of LittleBee® Family via DIN interface.
	CPU <sup>[5]</sup>	111	External Host configure FPGA products of LittleBee® Family via DBUS interface.

**Note!**

- [1] The unbound mode pins are grounded by default (except for GW1N(R)-2 and GW1N-1P5 devices, please refer to the corresponding pinout manuals).
- [2] The JTAG configuration mode is independent of MODE value.
- [3] The SPI interfaces of the SSPI and MSPI modes are independent of each other.
- [4] Currently GW1N(R)-4 / GW1N(R)-4B do not support DUAL BOOT.
- [5] The CPU configuration mode and SERIAL configuration mode share SCLK, WE\_N and CLKHOLD\_N. The data bus pins for the CPU configuration mode share pins with MSPI and SSPI configuration modes.
- [6] Autoboot is automatically enabled in I<sup>2</sup>C mode. In I<sup>2</sup>C mode, following power-on the LittleBee devices will attempt to read data from the internal Flash first. The I<sup>2</sup>C SDA line MUST be held inactive (externally pulled-up) during Autoboot, otherwise the device maynot be configured correctly. Also, it is recommended to externally pull up the SCL line at the same time. Note that this note also applies to C version devices of which the SDA and SCL pins are with internal weak pull-up.

**Note!**

For details about configuration pins, pin reuse, and pin functions and application, please refer to [4 Configuration Pins](#).

## 5.2 Arora Family of FPGA Products

Besides the JTAG configuration mode that is commonly used in the industry, the Arora Family of FPGA products also support GOWINSEMI's own configuration mode: GowinCONFIG. The GowinCONFIG configuration modes that are available and supported for each device depend on the device model and package. The Arora Family of FPGA Products support bitstream encryption and security bit setting, which provides safety for user designs. The Arora Family FPGA products support bitstream decompression; users can compress bitstream to save storage memory.

Table 5-2 lists the configuration modes that are supported by the Arora Family FPGA products.

**Table 5-2 Configuration Modes**

Configuration Modes		MODE[2:0] <sup>1</sup>	Description
JTAG		XXX <sup>2</sup>	External Host configures Arora Family of FPGA products via JTAG interface.
GowinCONFIG	MSPI	000	As Master, FPGA reads data from external Flash (or other devices) via the SPI interface <sup>3</sup> .
	SSPI	001	External Host configures Arora Family of FPGA products via SPI interface.
	SERIAL <sup>4</sup>	101	External Host configures Arora Family of FPGA products via DIN interface.
	CPU <sup>4</sup>	111	External Host configures Arora Family of FPGA products via DBUS interface.

**Note!**

- [1] The unbound mode pins are grounded by default.
- [2] The JTAG configuration mode is independent of MODE value.
- [3] The SPI interfaces of the SSPI and MSPI modes are independent of each other.
- [4] The CPU configuration mode and SERIAL configuration mode share SCLK, WE\_N and CLKHOLD\_N. The data bus pins for the CPU configuration mode share pins with MSPI and SSPI configuration modes.

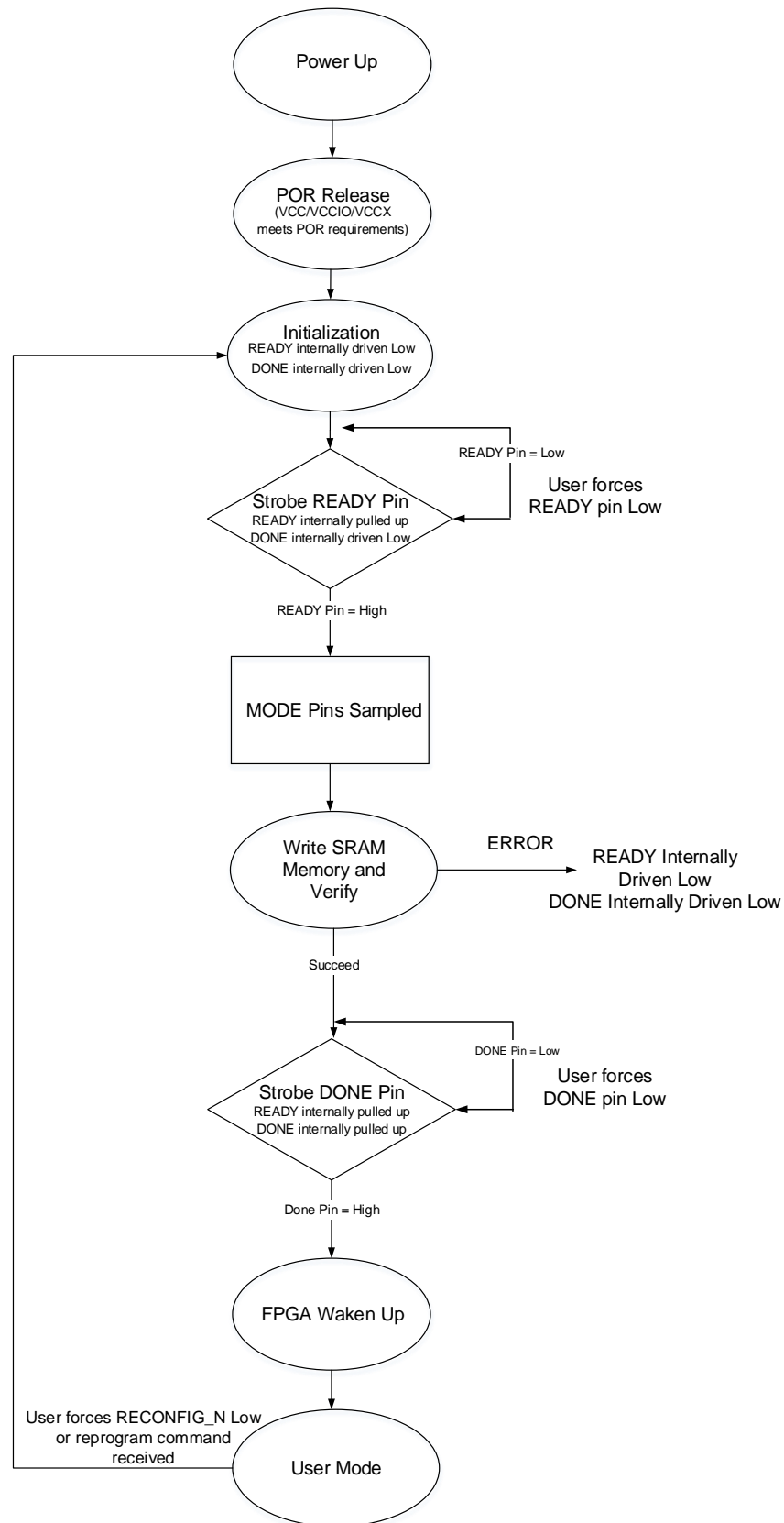
**Note!**

For details about configuration pins, pin reuse, and pin functions and application, please refer to [4 Configuration Pins](#).

# 6 Configuration Process

After power on, the FPGA goes through a sequence of states including initialization, SRAM configuration, and wake-up. The configuration flow is as shown in below.

Figure 6-1 Configuration Flow



**Note!**

- READY, DONE, and RECONFIG\_N are bidirectional IOs with open drain output and internal weak pull-up (the pull-up current is about 100uA).
- You can control the timing of the device starting to load by forcing the READY pin low.
- You can control the timing of the device waking up by forcing the DONE pin low.
- The RECONFIG\_N pin needs to be held high from power-up to full loading of the device.

## 6.1 Power-up Sequence

During the power-on process, the power-on reset (POR) circuit inside the FPGA becomes active. The active POR circuit makes sure the external I/O pins are in a high-impedance state and monitors the VCC/VCCX/VCCIO<sub>n</sub> input rails. When VCC/VCCX/VCCIO<sub>n</sub> meets the minimum reset voltage level (Voltage level may vary for different devices, and different devices monitor different power rails.), POR circuit releases an internal reset signal, allowing the FPGA to begin its initialization process. When READY and DONE are driven low, the FPGA moves to the initialization state, as shown in Figure 6-2.

Figure 6-2 POR Power-up Timing

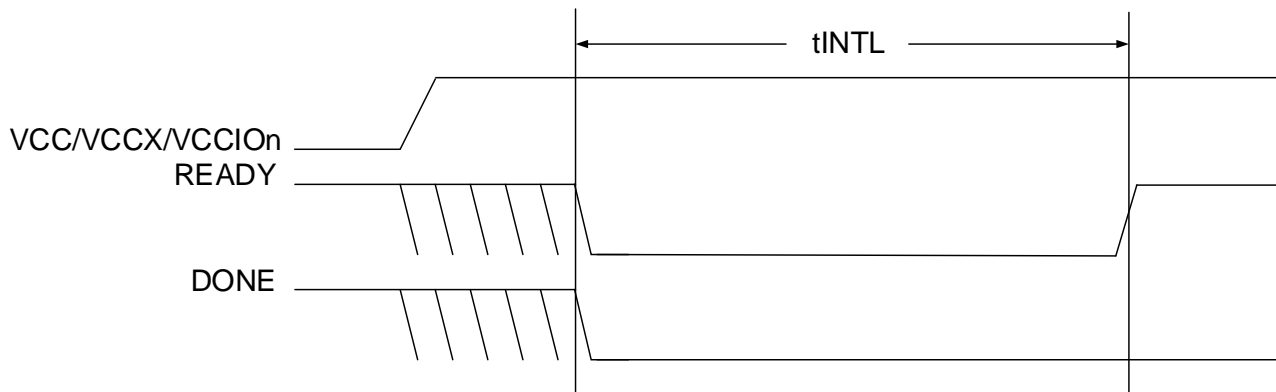


Table 6-1 lists different power rails monitored by POR circuits of different devices.

Table 6-1 Power Rails Monitored by POR Circuits of Different Devices

Series	Device	Power Rails
GW1N	GW1N-1 GW1N-4 GW1N-9	VCC/VCCX/VCCIO1/VCCIO3
	GW1N-1P5 GW1N-2	VCC/VCCX/VCCIO0
	GW1N-1S	VCC/VCCX/VCCIO0/VCCIO2
GW1NZ	GW1NZ-1	VCC/VCCX/VCCIO1/VCCIO3
GW1NR	GW1NR-1 GW1NR-2 GW1NR-4 GW1NR-9	VCC/VCCX/VCCIO1/VCCIO3
GW1NS	GW1NS-4 GW1NS-4C	VCC/VCCX/VCCIO0/VCCIO1
GW1NSR	GW1NSR-4	VCC/VCCX/VCCIO0/VCCIO1

Series	Device	Power Rails
	GW1NSR-4C	
GW1NSE	GW1NSE-4C	VCC/VCCX/VCCIO0/VCCIO1
GW1NSER	GW1NSER-4C	VCC/VCCX/VCCIO0/VCCIO1
GW1NRF	GW1NRF-4B	VCC/VCCX/VCCIO1/VCCIO3
GW2A	GW2A-18 GW2A-55	VCC/VCCX/VCCIO3
GW2AR	GW2AR-18	VCC/VCCX/VCCIO3
GW2AN	GW2AN-9X GW2AN-18X	VCC/VCCX/VCCIO1/VCCIO5
	GW2AN-55	VCC/VCCX/VCCIO3
GW2ANR	GW2ANR-18	VCC/VCCX/VCCIO3

## 6.2 Initialization

After the power on reset circuit drives the READY and DONE status pins low, the FPGAs enter the memory initialization immediately. The purpose of the initialization is to clear all the SRAM memory inside the FPGA.

The FPGA remains in the initialization state until all of the following conditions are met:

- The  $T_{INITL}$  time period has elapsed.
- The RECONFIG\_N pin is high.
- The READY pin is not driven low by an external driver.

The READY pin provides two functions during the initialization phase:

- To indicate that the FPGA is currently clearing its configuration SRAM
- To act as an input preventing the FPGA transition from the initialization state to the configuration state when it's driven low by an external driver.

## 6.3 Configuration

The rising edge of the READY pin causes the FPGA to enter the configuration state. The internal configuration SRAM of FPGA can be configured via multiple modes according to the MODE pin values. During the time the FPGA receives its configuration data, the READY pin can indicate its internal state. When READY is high, configuration proceeds without issue. If READY is low, an error has occurred and the FPGA does not operate.

## 6.4 Wake-up

When all the configuration data is received correctly, the FPGA enters the wake-up state and set the internal status bit of DONE to 1. In the wake-up state, the FPGA will perform the following operations in sequence:

1. Enable the Global output enable (GOE) signal, and then the FPGA I/O exits a high-impedance state and take on its programmed function. The input signals are prevented from performing any action on the FPGA flip-flops by the assertion of the Global Set/Reset (GSR).

2. Release the Global Set/Reset (GSR) signal and the Global Write Disable (GWDISn) signal. Enabling the Global Write Disable (GWDISn) signal prevents the FPGA from mistakenly overwriting the initialization data in the internal RAM.
3. Enable the external DONE pin. The external DONE is a bidirectional, open-drain I/O when it's enabled. Keep the FPGA wake-up by externally driving the DONE pin low. When the DONE pin is driven high, the FPGA wake-up phase is complete and enters user mode.

## 6.5 User Mode

After entering user mode, the FPGA will perform the logic operations you designed immediately. The FPGA will remain in this state until one of the following three events occurs:

- The RECONFIG\_N pin is externally driven low.
- A reprogram command is received via one of the configuration ports
- Power is cycled

Once one of the three events above occurs, the FPGA will enter the configuration process again.

# 7 Configuration Mode Details

Gowin FPGA products include the SRAM-based high-performance Arora Family of FPGA products and small capacity nonvolatile device of the LittleBee® Family of FPGA products with embedded Flash. Any configuration data that is stored in the SRAM device is lost after it is powered down; as such, it needs to be reconfigured each time it is powered up. The data stored in non-volatile devices with built-in flash is still stored in the chip if the device is powered down, and the device can be automatically reconfigured after power up via the AUTOBOOT or DUALBOOT configuration options.

Gowin FPGA products have abundant packages. The configuration modes supported by each device are related to the number of configuration pins bonded out: All devices support JTAG configuration, but only non-volatile devices support AUTO BOOT or DUAL BOOT configuration. The mode value for each configuration is different.

## 7.1 Configuration Notes

GOWINSEMI FPGA products include LittleBee® family and Arora family. Whether the name of the device contains R does not affect the configuration feature, the main difference is that SDRAM/PSRAM is integrated in all FPGA products that have a serial number that includes the letter R. Except DUALBOOT configuration features, the GW1NS series of FPGA products have same features as the GW1N series.

### Power Up and Configuration Flow

When the power up voltage of VCC, VCCIO, and VCCX reaches the min. value, FPGA begins to start: stable voltage and RECONFIG\_N is not pulled down > The internal circuit of FPGA pulls down READY and DONE pins > FPGA initialization > Pulling up READY and sampling MODE value > Reading and checking the configuration data according to the configuration mode > FPGA waking up > DONE pulling up > Entering user mode.

The power supply needs to be stable during FPGA startup. No low levels are allowed on the RECONFIG\_N pin during power-up, initialization,

and configuration of the FPGA. You can choose to leave the RECONFIG\_N pin floating or pull up the RECONFIG\_N pin externally. From the release of Power-on Reset to just before the wake-up of the device, all GPIOs are high-impedance with internal weak pull-ups.

GOWINSEMI FPGA products write bitstream data to SRAM, on-chip Flash, or off-chip Flash according to the data storage and the instructions. Only the LittleBee® Family of FPGA products support operations on on-chip Flash. All products support operations on SRAM and external Flash.

### SRAM Operation

The SRAM operations include read device ID CODE and USER CODE, read device status register information and SRAM configuration. The device ID needs to be verified before configuration. Only the device with successful ID verification can be configured. The USER CODE is the identification number for users to distinguish between the devices that share the same ID CODE. The state register of the device records the status information before and after FPGA configuration, and you can use this information to analyze the state of the device accordingly. Please refer to Table 7-12 for the meaning of the status register. During SRAM configuration, only the bitstream data with no security bit setting supports validation. Data with security bit cannot be readback or verified.

### On-chip/Off-chip Flash Operation

The built-in flash operations include erasing, programming and verification. The built-in flash can only be programmed via the JTAG interface, and the clock rate is no less than 1MHz. Please refer to Table 7-9 for the clock rate.

#### **Note!**

During configuring SRAM devices via built-in Flash (AUTOBOOT configuration and DUALBOOT configuration) and programming built-in Flash, the FPGA needs to remain powered up, and the RECONFIG\_N cannot be triggered at low level; otherwise, it may cause irreparable damages to the built-in Flash.

The LittleBee® family devices (except the GW1N-4 A version) support the feature of background upgrade. That is to say, you can program the embedded Flash or external Flash via the JTAG<sup>[1]</sup> interface without affecting the current working state. During programming, the device works according to the previous configuration. After programming, RECONFIG\_N is triggered at low pulse to complete the online upgrade. This feature applies to the applications requiring long online time and irregular upgrades.

#### **Note!**

[1] GW1N-1P5 and GW1N-2 can support the I<sup>2</sup>C background upgrade by using the goConfig I2C IP. However, it is recommended to use the JTAG interface to implement the background upgrade.

## Dual-purpose Pin Configuration

In different configuration modes, users need to ensure that FPGA works in the selected configuration mode according to the pin functions. If user pins is insufficient, these pins can be configured and used as GPIOs, but pins associated with data transmission need to be kept. MODE [2:0] is used to select the GowinCONFIG programming configuration mode. The configuration mode can be fixed by using pull-up or pull-down resistors. It is recommended to use a 4.7K resistor for pull-up or a 1K resistor for pull-down.

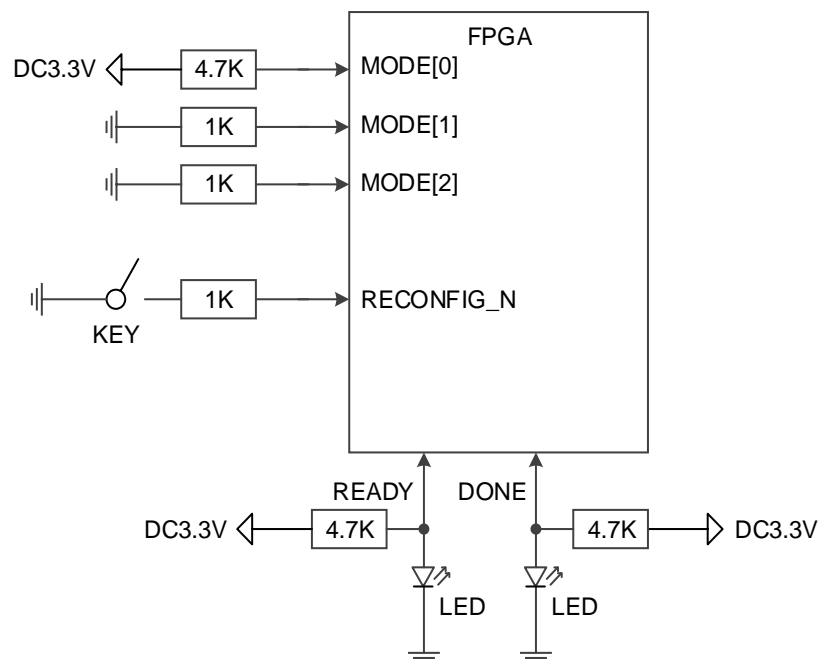
### Note!

The RECONFIG\_N, READY, and DONE pins are associated with each configuration mode. Whether they are set as GPIO or not, users should ensure that their initial value or pin connection state meets programming and configuration conditions before completing the configuration process.

## Recommended Pin Connection

When users are designing a circuit schematic diagram, the recommended connection is as shown in Figure 7-1.

Figure 7-1 Recommended Pin Connection



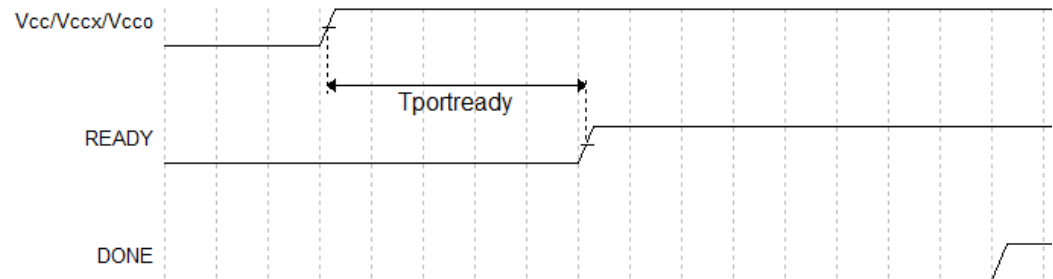
### Note!

- You can add a DIP switch to change the MODE value. For some devices where the MODE pins are not all bonded out, the unbonded MODE pins are grounded by default (except for GW1N(R)-2 and GW1N-1P5 devices, please refer to the corresponding pinout manuals).
- The values of READY and DONE signals have no meaningful reference in JTAG configuration.
- The unbonded RECONFIG\_N, READY, and DONE pins have been internally processed, with no influence on the configuration function.

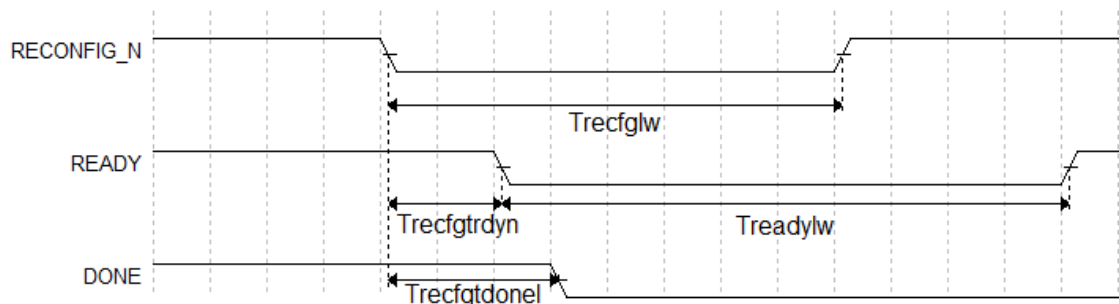
### Timing for Power Cycling and RECONFIG\_N Triggering at Low Pulse

Figure 7-2 and Figure 7-3 show the timing for power cycling and RECONFIG\_N triggering at low pulse.

**Figure 7-2 Power Cycling Timing**



**Figure 7-3 RECONFIG\_N Triggering Timing**



Timing parameters of the LittleBee® Family of FPGA Products is shown in Table 7-1.

**Table 7-1 Timing Parameters for Power Cycling and RECONFIG\_N Triggering(LittleBee® Family)**

Name	Description	Min.	Max.
$T_{portready}^1$	Time from POR to the rising edge of READY	50μs	200μs
$T_{recfglw}$	RECONFIG_N low pulse width	25ns	-
$T_{recfgtrdyn}$	Time from RECONFIG_N falling edge to READY low	-	70ns
$T_{readylw}$	READY low pulse width	TBD	-
$T_{recfgtdonel}$	Time from RECONFIG_N falling edge to READY low	-	80ns

**Note!**

In the case of MODE0=0, the device power-up waiting time is 200 μs; If MODE0=1, the device power-up waiting time is 50 μs.

Timing parameters of the Arora Family of FPGA Products are as shown in Table 7-2.

**Table 7-2 Timing Parameters for Power Cycling and RECONFIG\_N Triggering(Arora Family)**

Name	Description	Min.	Max.
T <sub>portready</sub>	Time from POR to the rising edge of READY	-	35ms
T <sub>recfglw</sub>	RECONFIG_N low pulse width	25ns	-
T <sub>recfgtrdyn</sub>	Time from RECONFIG_N falling edge to READY low	-	70ns
T <sub>readylw</sub>	READY low pulse width	TBD	-
T <sub>recfgtdonel</sub>	Time from RECONFIG_N falling edge to READY low	-	80ns

## 7.2 JTAG Configuration Mode

The JTAG configuration mode of Gowin FPGA products conforms to the IEEE1532 standard and the IEEE1149.1 boundary scan standard.

The JTAG configuration mode writes bitstream data to the SRAM of Gowin FPGA products. All configuration data is lost after the device is powered down. All Gowin FPGA products support the JTAG configuration mode.

### 7.2.1 JTAG Configuration Mode Pins

The relevant pins for the JTAG configuration mode are shown in Table 7-3.

**Table 7-3 Pin Description in JTAG Configuration Mode**

Pin Name	I/O	Description
JTAGSEL_N <sup>[1]</sup>	I, internal weak pull-up	Revert JTAG pin from GPIO to configuration pin. Low active
TCK <sup>[2]</sup>	I	JTAG serial clock input
TMS	I, internal weak pull-up	JTAG serial mode input
TDI	I, internal weak pull-up	JTAG serial data input
TDO	O	JTAG serial data output

**Note!**

- [1] The JTAGSEL\_N works only when the JTAG pin is set as a GPIO and the device starts to work. For the LittleBee® Family of FPGA products, when MODE[2: 0]=001, the JTAGSEL\_N pin is always a GPIO, in other words the JTAGSEL\_N pin and the four JTAG pins (TCK, TMS, TDI, TDO) can be used as GPIOs simultaneously; however, in this case the JTAGSEL\_N pin cannot restore the JTAG pins to configuration IOs, and these pins will be restored as configuration IOs when the FPGA re-enters edit mode.
- [2] TCK needs to connect 4.7 K pull down resistor on the PCB.

For some devices, if the four pins of JTAG or JTAGSEL\_N are multiplexed as GPIOs, a reprogram instruction is required if you want to reconfigure the device. The details are shown in Table 7-4.

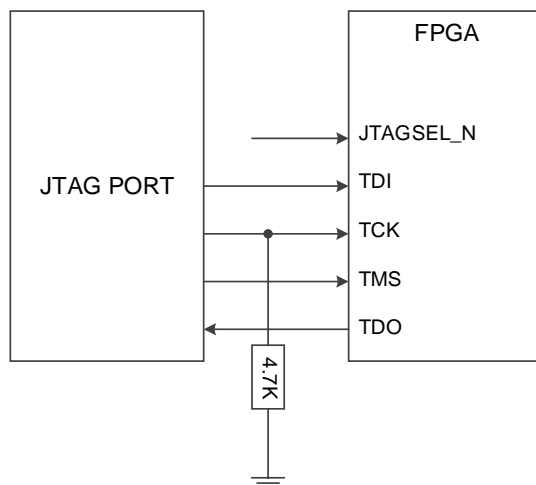
**Table 7-4 List of devices for which you need/do not need to send a reprogram instruction.**

Series	Device	need to send a reprogram instruction?
GW1N	GW1N-1, GW1N-1S, GW1N-4, GW1N-4B, GW1N-4D, GW1N-9, GW1N-9C,	YES
GW1NZ	GW1NZ-1, GW1NZ-1C,	
GW1NR	GW1NR-1, GW1NR-4, GW1NR-4B, GW1NR-4D, GW1NR-9, GW1NR-9C	
GW1NRF	GW1NRF-4B	
GW2A	GW2A-18, GW2A-18C, GW2A-55C, GW2A-55	
GW2AR	GW2AR-18, GW2AR-18C	
GW2AN	GW2AN-55C	
GW2ANR	GW2ANR-18C	
GW1N	GW1N-1P5, GW1N-1P5B, GW1N-1P5C, GW1N-2, GW1N-2B, GW1N-2C	NO
GW1NS	GW1NS-4, GW1NS-4C	
GW1NR	GW1NR-2, GW1NR-2B, GW1NR-2C	
GW1NSR	GW1NSR-4C, GW1NSR-4	
GW1NSER	GW1NSER-4C	

## 7.2.2 Connection Diagram for the JTAG Configuration Mode

The connection diagram in the JTAG configuration mode is shown in Figure 7-4.

Figure 7-4 Connection Diagram for JTAG Configuration Mode

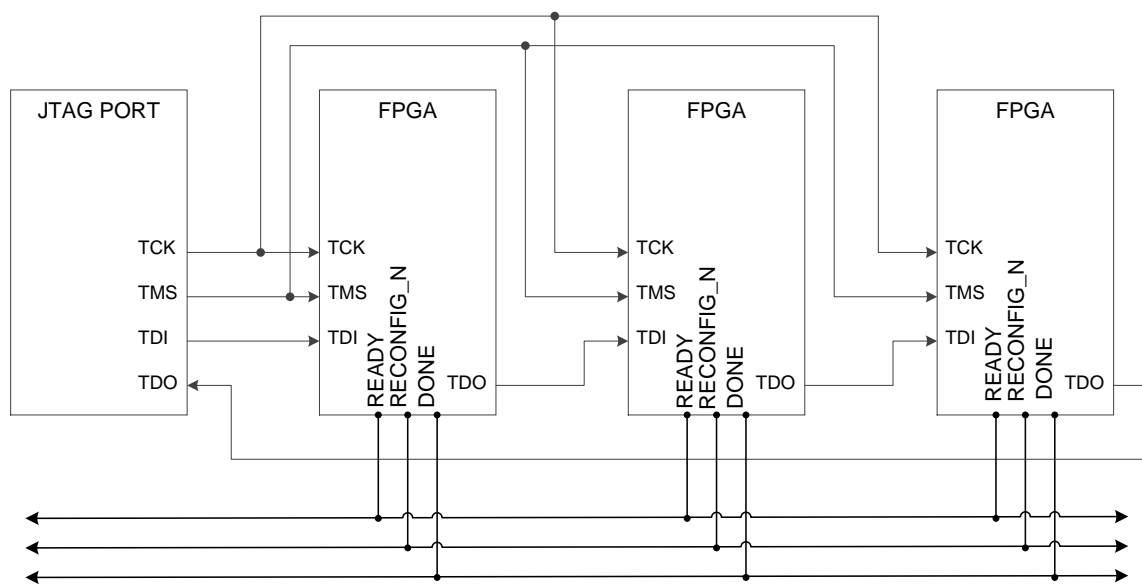


### Note!

- If JTAGSEL\_N is not bonded out, when debugging the JTAG pin reuse, it is suggested to set the MODE value to non-auto configuration mode (AUTOBOOT, DUALBOOT or MSPI) before powering up the device to avoid the other bitstream data affecting configuration. After power up and JTAG is configured manually, the device enters User MODE, and JTAG pin will be used as a GPIO.
- The clock frequency for JTAG configuration mode is no higher than 40MHz.

In addition to using JTAG to configure SRAM, the built-in Flash of Gowin non-volatile FPGA devices (LittleBee® Family) and the external SPI Flash of all other FPGA series programming can also be configured through the JTAG pin. The connection for programming the built-in Flash of the non-volatile devices is the same as that of the JTAG mode. Please refer to Figure 7-51 and 9 Boundary Scan for external SPI Flash programming.

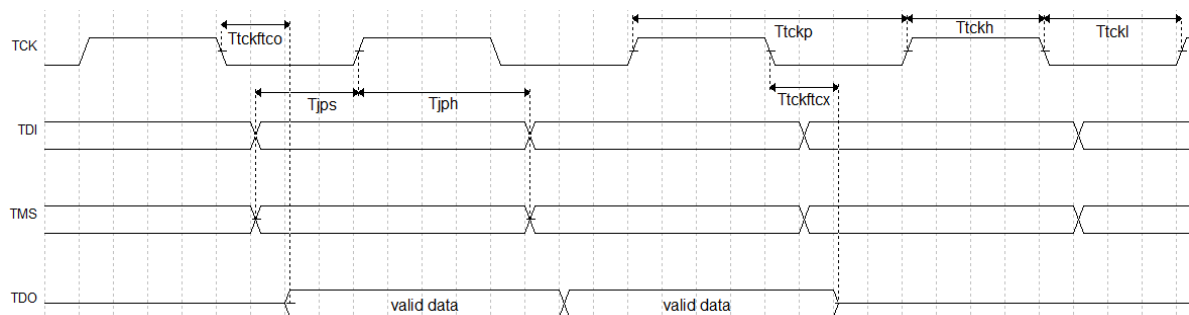
In addition, Gowin FPGA products support JTAG daisy chain operation, which connects the FPGA TDO pin to the next FPGA TDI pin. Gowin programming software will identify the connected FPGA devices automatically and configure them in turn. The connection diagram for the daisy chain configuration is shown in Figure 7-5.

**Figure 7-5 Connection Diagram of JTAG Daisy-Chain Configuration Mode****Note!**

DONE, RECONFIG\_N, and READY can be connected or not as appropriate.

## 7.2.3 JTAG Configuration Timing

See Figure 7-6 for the timing of JTAG mode.

**Figure 7-6 JTAG Configuration timing**

See Table 7-5 for the description of timing parameters.

**Table 7-5 JTAG Configuration Timing Parameters**

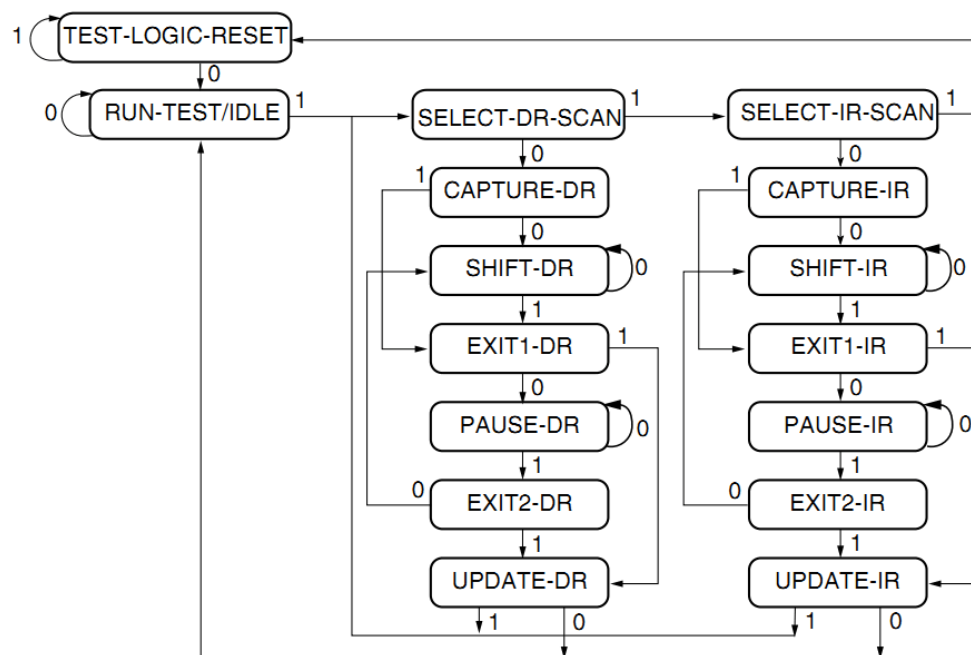
Name	Description	Min.	Max.
$T_{tckftco}$	Time from TCK falling edge to output	-	10ns
$T_{tckftcx}$	Time from SCLK falling edge to high impedance	-	10ns
$T_{tckp}$	TCK clock period	40ns	-
$T_{tckh}$	TCK clock high time	20ns	-
$T_{tckl}$	TCK clock low time	20ns	-
$T_{jps}$	JTAG PORT setup time	10ns	-
$T_{jph}$	JTAG PORT hold time	8ns	-

## 7.2.4 JTAG Configuration Process

### TAP State Machine

The state machine for the test access port is designed to select an instruction register or a data register to connect it between TDI and TDO. In general, the instruction register is used to select the data register to be scanned. In the state machine diagram, the number on the side of the arrow indicates the logic state of the TMS when the TCK goes high, as shown in Figure 5-7.

Figure 7-7 TAP State Machine



### TAP Reset

After TMS keeps high (logic "1") and at least 5 strobes are input (higher and then low) at the TCK terminal, the TAP logic is reset, the TAP state machine in other states is converted into the state of test logic reset, and the JTAG port and the test logic are reset.

#### Note!

The CPU and peripherals are not reset in this state.

#### Note!

- The data on the TDO is valid from the falling edge of TCK in the Shift\_DR or Shift\_IR state.
- The data is not shifted in the Shift\_DR or Shift\_IR state.
- The data is shifted when leaving the Shift\_DR or Shift\_IR.
- The first to be shifted is the least significant bit (LSB) of the data.
- Once reset, all instructions will be reset or disabled.

### Instruction Register and Data register

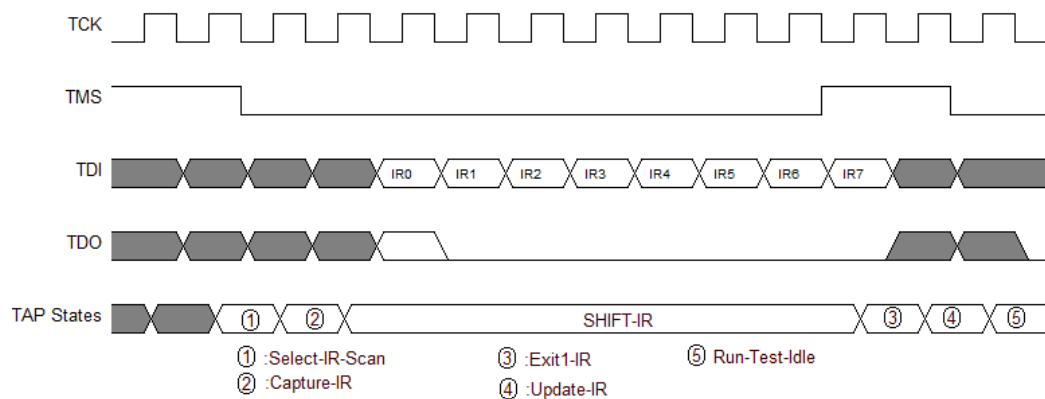
In addition to the test logic reset, the state machine can also control two basic operations:

- Instruction register (IR) scan.
- Data Register (DR) scan.

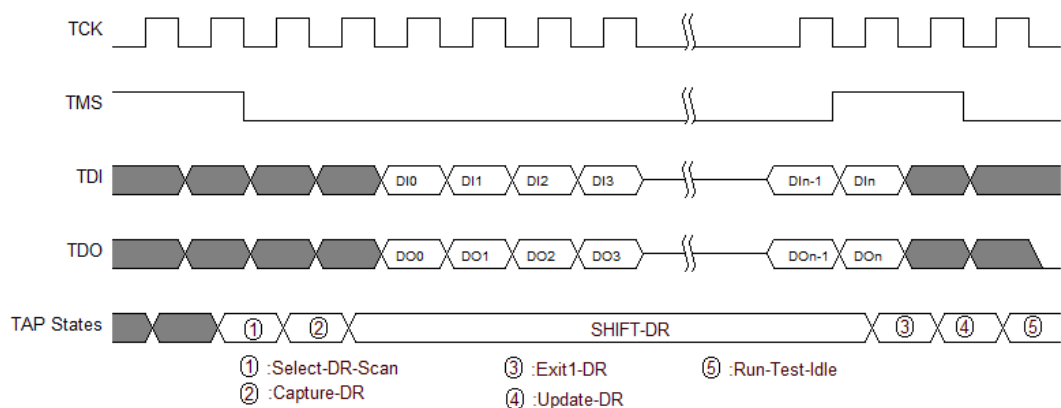
During the IR scanning operation, in Shift\_IR state, the data or instructions are sent to the IR in the LSB way. The lower data bits are sent first. The instructions will be all sent when the state machine returns to Run-Test-Idle, as shown in Figure 7-8

During the data register scanning operation, the data or instructions are sent to the DR in the Shift\_DR state, as shown in Figure 7-9. The data is sent in LSB way or MSB way depending on specific operations.

**Figure 7-8 Instruction Register Access Timing**



**Figure 7-9 Data Register Access Timing**



**Note!**

- The total length of the instruction register is 8 bits in the GW1N(R) and GW2A(R) series of the FPGA.
- The length of the data register can vary depending on the selected register.

**Read ID CODE Instance**

ID Code, i.e. JEDEC ID Code, is a basic identification of FPGA products.

The length of the Gowin FPGA ID Code is 32 bits. The ID Codes of the FPGA are listed in the following table.

**Table 7-6 Gowin FPGA ID CODE**

Gowin FPGA Device Family ID CODE			
Device Family	Device Part	Manufacturer ID	ID CODE
	Bits 31-12	Bits 11-0 h81B	
GW1N-1	h09002	h81B	h0900281B
GW1N-1S	h09003		h0900381B
GW1NZ-1	h01006		h0100681B
GW1N-2/2B	h01206		h0120681B
GW1N-1P5/1P5B	h01206		h0120681B
GW1N(R)-4	h01003		h0100381B
GW1N(R)-4B	h11003		h1100381B
GW1N(R)-4D	h11003		h1100381B
GW1NS(ER)-4C	h01009		h0100981B
GW1N(R)-9	h11005		h1100581B
GW1N(R)-9C	h11004		h1100481B
GW2A(R)-18/18C	h00000		h0000081B
GW2A-55/55C	h00002		h0000281B

The instruction for reading FPGA is 0x11. Take the GW1N-4B ID Code as an example to illustrate the working mode of JTAG, please refer to the following steps:

1. TAP reset: TMS is set to high level and at least 5 clock cycles are continuously transmitted.
2. Move the state machine from Test-Logic-Reset to Run-Test-Idle.
3. Move the state machine to Shift-IR. Send Read ID instruction (0x11) beginning with LSB. When MSB (the last bit) is being sent, move state machine to Exit1-IR at the same time, i.e., TMS should be high level before sending MSB. Table 7-7 shows the change of TDI and TMS value during sending 0x11 in 8-clock cycle. The timing is as shown in Figure 7-11.

**Table 7-7 Change of TDI and TMS Value in The Process of Sending Instructions**

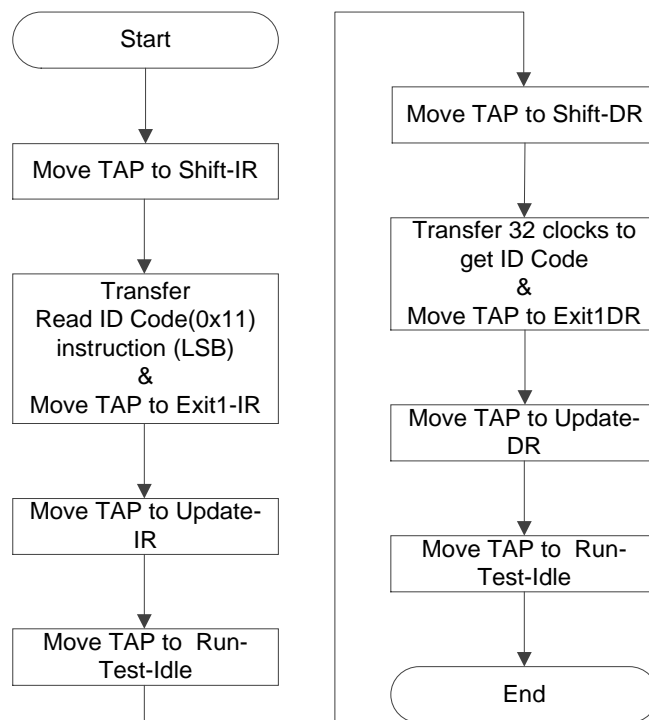
	TCK 1	TCK 2	TCK 3	TCK 4	TCK 5	TCK 6	TCK 7	TCK 8
TDI value (0x11)	1	0	0	0	1	0	0	0
TMS value	0	0	0	0	0	0	0	1

4. Move the state machine, back to Run-Test-Idle after going from Exit1-IR to Update-IR, and then run the state machine at least 3 clock cycles in Run-Test-Idle.
5. Move the state machine to Shift-DR, send 32 clock cycles, and set TMS to high level before the 32nd clock is sent. When the 32 clock cycles are completed, jump from Shift-DR to Exit1-DR. During this period, sending 32 clocks can read 32 bits data, that is, 0x0100381B,

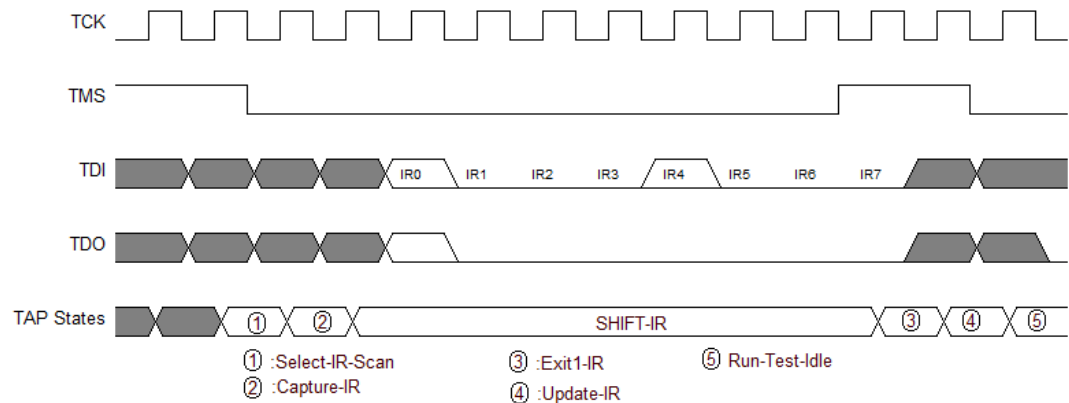
as shown in Figure 7-12.

6. Move the state machine back to Run-Test-Idle.

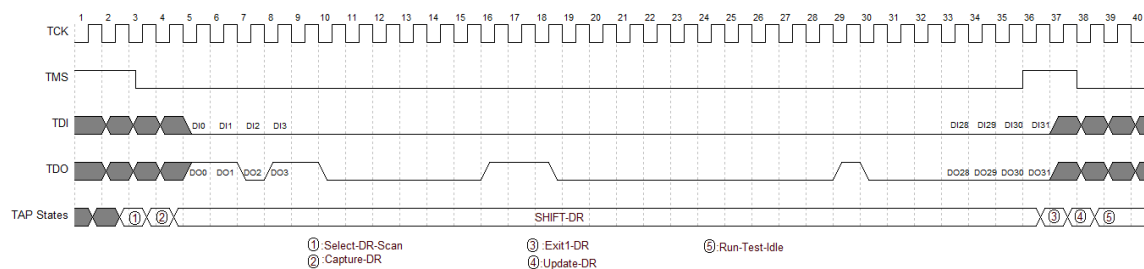
**Figure 7-10 State Machine Flowchart of Reading ID Code**



**Figure 7-11 Instruction-0x11 Access Timing When Reading ID Code**



**Figure 7-12 Data Register Access Timing When Reading ID Code**

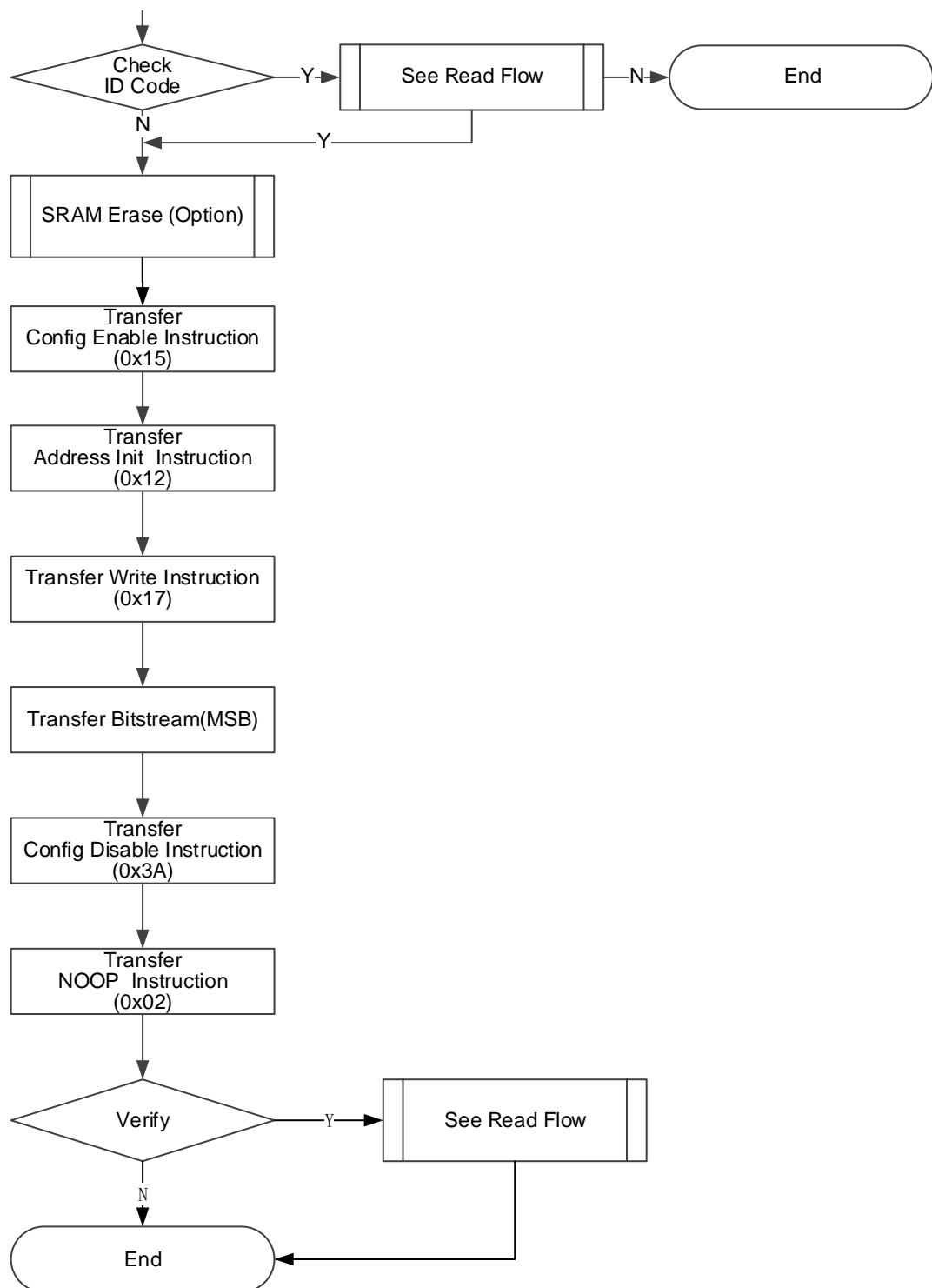


### Process of SRAM Configuration

The FPGA SRAM is configured using an external Host to enable the FPGA functions. SRAM is configured via JTAG to avoid the influence of Configuration Mode Pins.

Generate the FS file using Gowin Software. Configure SRAM via JTAG. The process of SRAM configuration using the external Host is as follows, as shown in Figure 7-13.

1. Establish a JTAG link and reset TAP.
2. Read the device ID CODE and check if it matches.
3. Erase the SRAM if it has been configured. Please refer to "[SRAM Erasure Process](#)".
4. Send the "0x15" instruction of ConfigEnable.
5. Send the "0x12" instruction of Address Initialize.
6. Send the "0x17" instruction of Transfer Configuration Data.
7. Move the state machine to Shift-DR (Data Register). Send Configuration Data from the MSB bit by bit till all the bitstream file content is sent.
8. Send the "0x3A" instruction of Config Disable.
9. Send the "0x02" instruction of Noop to end the configuration process.
10. Please refer to Process of Reading SRAM (The process of reading SRAM) if reading back Configuration Data is required for verification.

**Figure 7-13 SRAM Configuration Flow****Process of Reading SRAM**

Warning: SRAM data is not allowed to be read back by default.

Read the SRAM data from the SRAM area of the FPGA. First ensure that the security bit is not configured when the data are written to the SRAM. The security bit is used to protect the runtime data and ensure the data security. After the safety bit is set, the data received from the SRAM are 1 (high level).

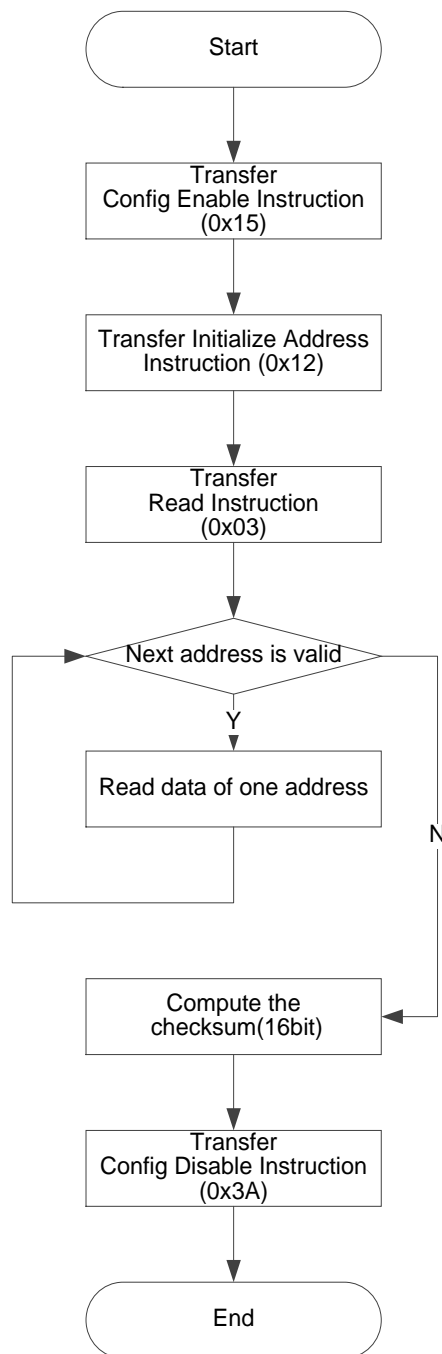
During loading, FPGA performs CRC check on the written data to ensure that the data is written correctly, and whether CRC reports an error can be used as a check mechanism to configure SRAM.

**Table 7-8 Count of Address and Length of One Address**

Device	Length of One Address (bits/address)	Count of Address
GW1N-1/GW1N-1S/ GW1NZ-1/GW1NR-1	1216	274
GW1N-1P5/GW1N-2/ GW1NR-2	1216	466
GW1N(R)-4/GW1NS(R)- 4/GW1NS(R)- 4C/GW1NSE(R)- 4C/GW1NRF-4B	2296	494
GW1N(R)-9	2836	712
GW2A(R)-18/GW2ANR- 18	3376	1342
GW2A(R)- 55(ES)/GW2AN-55	5536	2038

The reading process is described in detail below, as shown in Figure 7-14.

1. Send the "0x15" instruction of ConfigEnable.
2. Send the "0x12" instruction of Address Initialize.
3. Send the "0x 03" instruction of SRAM Read.
4. Move the state machine to Shift-DR (data register) and send as many clocks as the value of the address length, see Table 7-8. When the last clock is sent, pull up TMS at the same time. The state machine jumps to Exit1-DR, and TDO reads data with corresponding length. The state machine will return to Run-Test-Idle state finally.
5. Repeat the step 4, the address will be automatically accumulated when the data of an address are read each time.
6. Send the "0x3A" instruction of Config Disable.
7. Send the "0x02" instruction of Noop to end the reading process.

**Figure 7-14 Process of reading SRAM****Process of Erasing SRAM**

When reconfiguring SRAM, the existing SRAM needs to be erased. The flow is as follows:

1. Send the "0x15" instruction of ConfigEnable.
2. Send the "0x05" instruction of SRAM Erase.
3. Send the "0x02 " instruction of Noop.
4. Delay or Run Test 2~10ms.
5. Send the "0x09" instruction of SRAM Erase Done.
6. Send the "0x3A" instruction of Config Disable.

7. Send the "0x02" instruction of Noop to end the Erasure process.

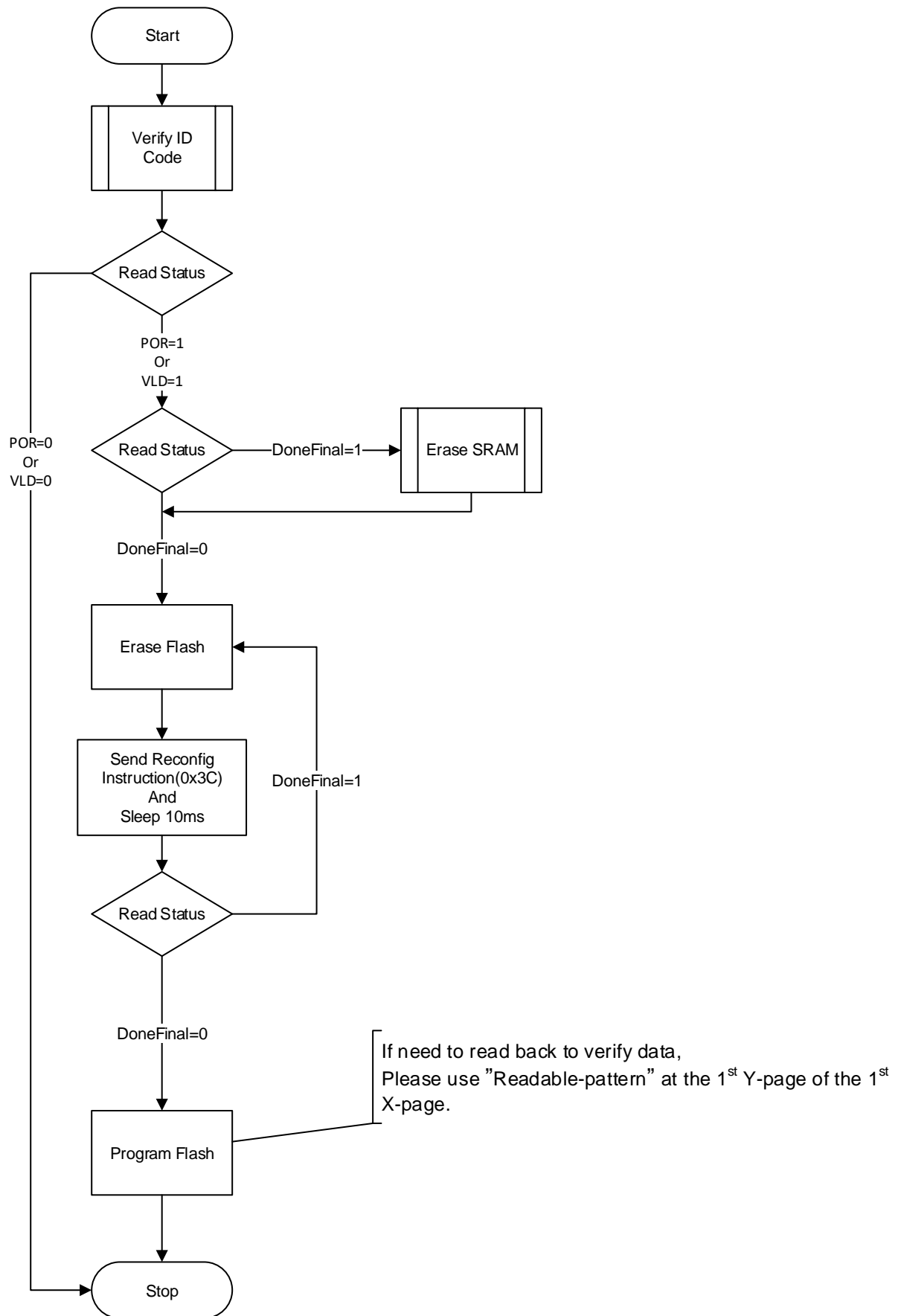
**Note!**

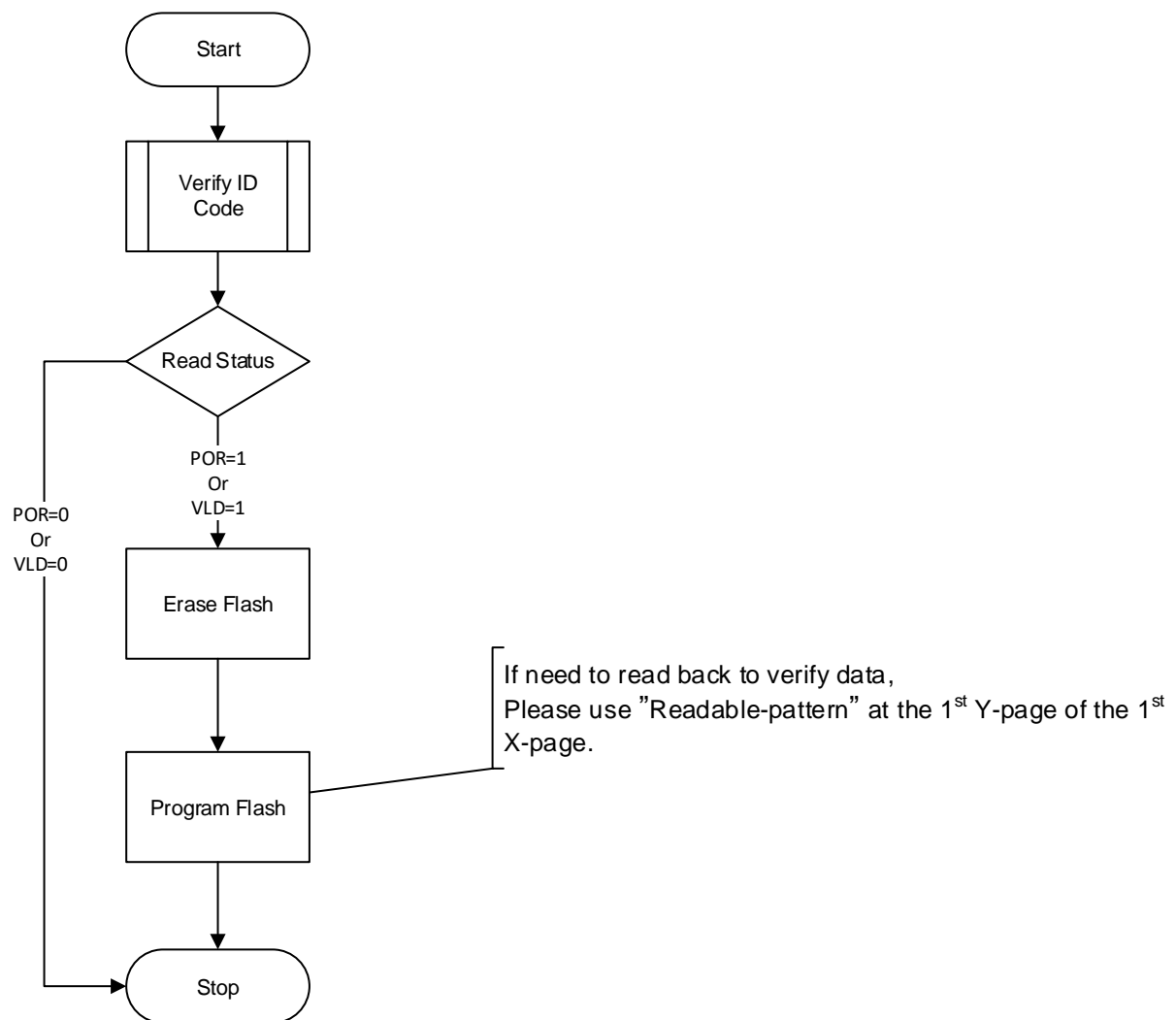
You need to wait enough time for the device to finish erasing after the instructions of EraseSram(0x05) and Noop(0x02) are sent.

- The reference time for GW1N(\*)-1 is 1ms.
- The reference time for GW1N(\*)-4 is 2ms.
- The reference time for GW1N(\*)-9 is 4ms.
- The reference time for GW2A(\*)-18 is 6ms.
- The reference time for GW2A(\*)-55 is 10ms.

**Process of Programming Internal Flash**

Programming the internal flash includes normal programming and background programming. Show the programming flows.

**Figure 7-15 Process of Normal Programming**

**Figure 7-16 Process of Background Programming**

### Erasing Internal Flash

For the embedded Flash memory of GW1N series, the embedded Flash needs to be erased before each programming task. For data security, the embedded Flash must be erased entirely.

The requirements for JTAG programming frequency are different according to the different processes of the GW1N series of the embedded Flash. Please refer to Table 7-9.

**Table 7-9 TCK Frequency Requirements for JTAG**

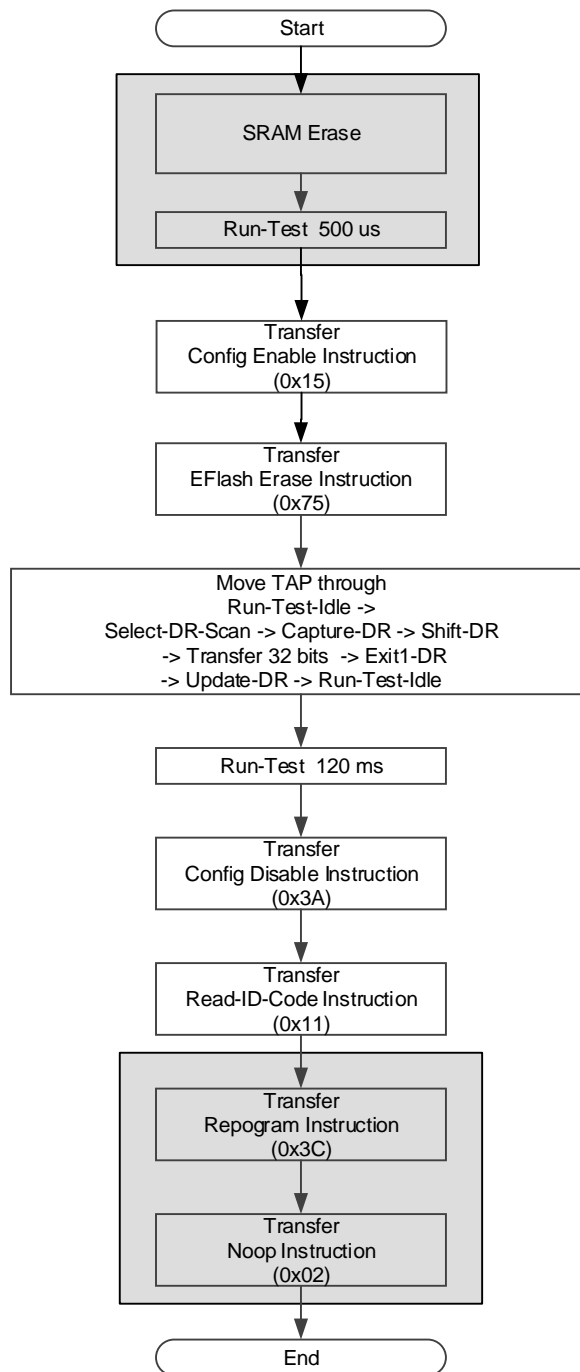
Device	TCK Frequency Range	Process Code
GW1N-1 GW1N-1S	1.4MHz ~ 5MHz	H
GW1N-2, GW1N-1P5	1.3MHz ~ 30MHz	T
GW1N(RF)-4B GW1NSER-4C GW1N(R)-9(C) GW1NZ-1	1.3MHz ~ 30MHz	T
GW2AN-55	0MHz ~ 25MHz	-

Device	TCK Frequency Range	Process Code
GW2ANR-18	0MHz ~ 40MHz	-

### Process of Erasing T-process FPGAs

The following describes the process of erasing the T-process FPGA(GW1NZ-1) in detail, as shown in Figure 7-17.

1. Establish a JTAG link and reset the TAP.
2. Read the device ID CODE and check if it matches.
3. Erase SRAM first if it has been configured.
4. The clock (Run-Test) is continuously generated in Run-Test-Idle for 500 $\mu$ s.
5. Send the "0x15" instruction of ConfigEnable.
6. Send the "0x75" instruction of EFlash Erase.
7. Move the state machine in turn: Run-Test-Idle -> Select-DR-Scan-> Capture-DR -> Shift-DR -> Transfer 32 bits-> Exit1-DR -> Update-DR -> Run-Test-Idle.
8. The clock (Run-Test) is continuously generated in Run-Test-Idle for 120ms. Please refer to Table 7-9 for the frequency requirements.
9. Send the "0x3A" instruction of Config Disable.
10. Send the "0x02" instruction of Noop to end the erasure process.
11. Send the "0x03" instruction of Reprogram to reconfigure the device and check if it erases successfully.

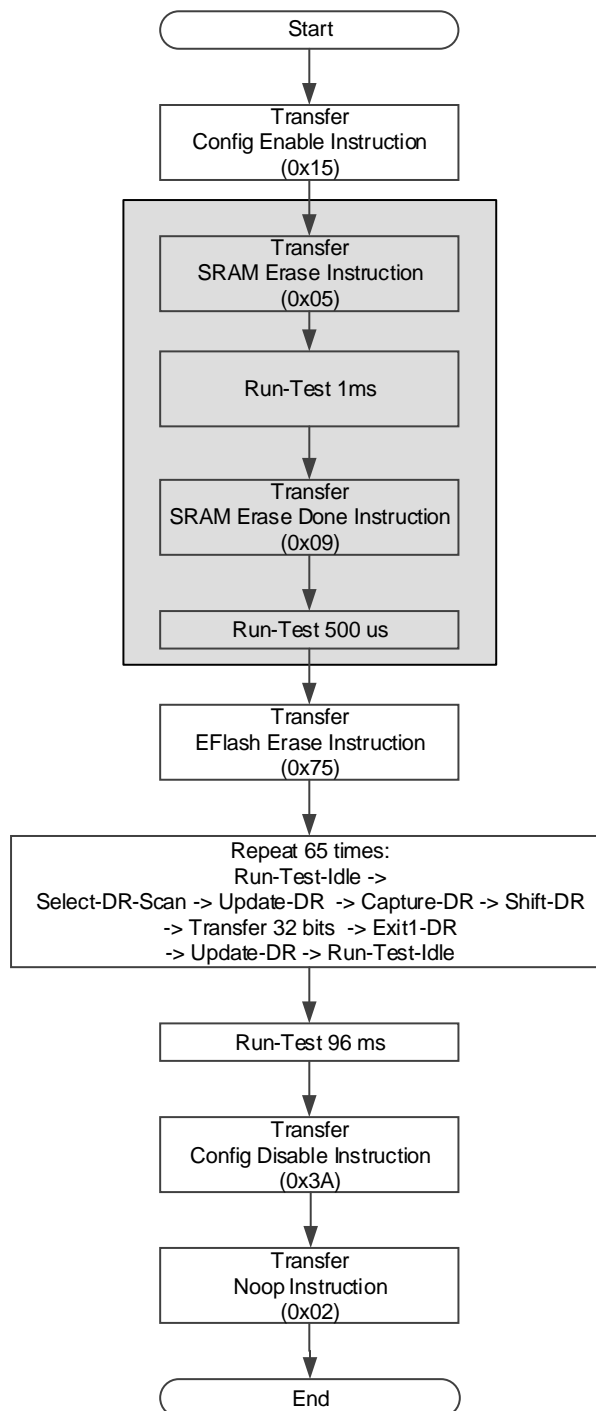
**Figure 7-17 Process of Erasing T-process FPGAs****Note!**

Ignore the shading area operation during Background Programming.

**Process of Erasing H-process FPGAs**

Process of Erasing H-process FPGAs:

1. Send the "0x15" instruction of ConfigEnable.
2. Send the "0x75" instruction of EFlash Erase.
3. Move the state machine from Run-Test-Idle to Shift-DR; 32 clocks are generated (TDI signal keeps low level). Move the state machine to Exit1-DR at the 32th clock, and then return to Run-Test-Idle going from Update-DR.
4. Repeat the steps above, 65 times in all.
5. The clock (Run-Test) is continuously generated in Run-Test-Idle for 120ms. Please refer to Table 7-9 for the frequency requirements.
6. Send the "0x3A" instruction of Config Disable.
7. Send the "0x03" instruction of Reprogram to check if the erasing is successful.
8. Send the "0x02" instruction of Noop to end the erasure process.

**Figure 7-18 Erase Flow for H-process FPGAs**

### Process of Programming Internal Flash

The internal Flash uses 256Bytes as an X-page. Each X-page is divided into 64 Y-pages, and each Y-page contains 4Bytes.

The first Y-page of the first X-page is used to identify whether the Flash has the capability of Autoboot or Readback, as shown in Table 7-10. When Readable-pattern is written to the first Y-page, the Flash data can be read; when the Autoboot-pattern is written to the first Y-page, the device automatically loads the Flash data into the SRAM in the autoboot mode;

The Flash can only be read after the Readable-pattern is written. It cannot be read in any other cases. Devices with the feature of background programming just need to use Autoboot-pattern.

Autoboot-pattern data must be inserted in the header of bitstream file in the case of no requirements of reading back data. If an X-page is less than 256Bytes, you can use 0xFF or 0x00 to complement it.

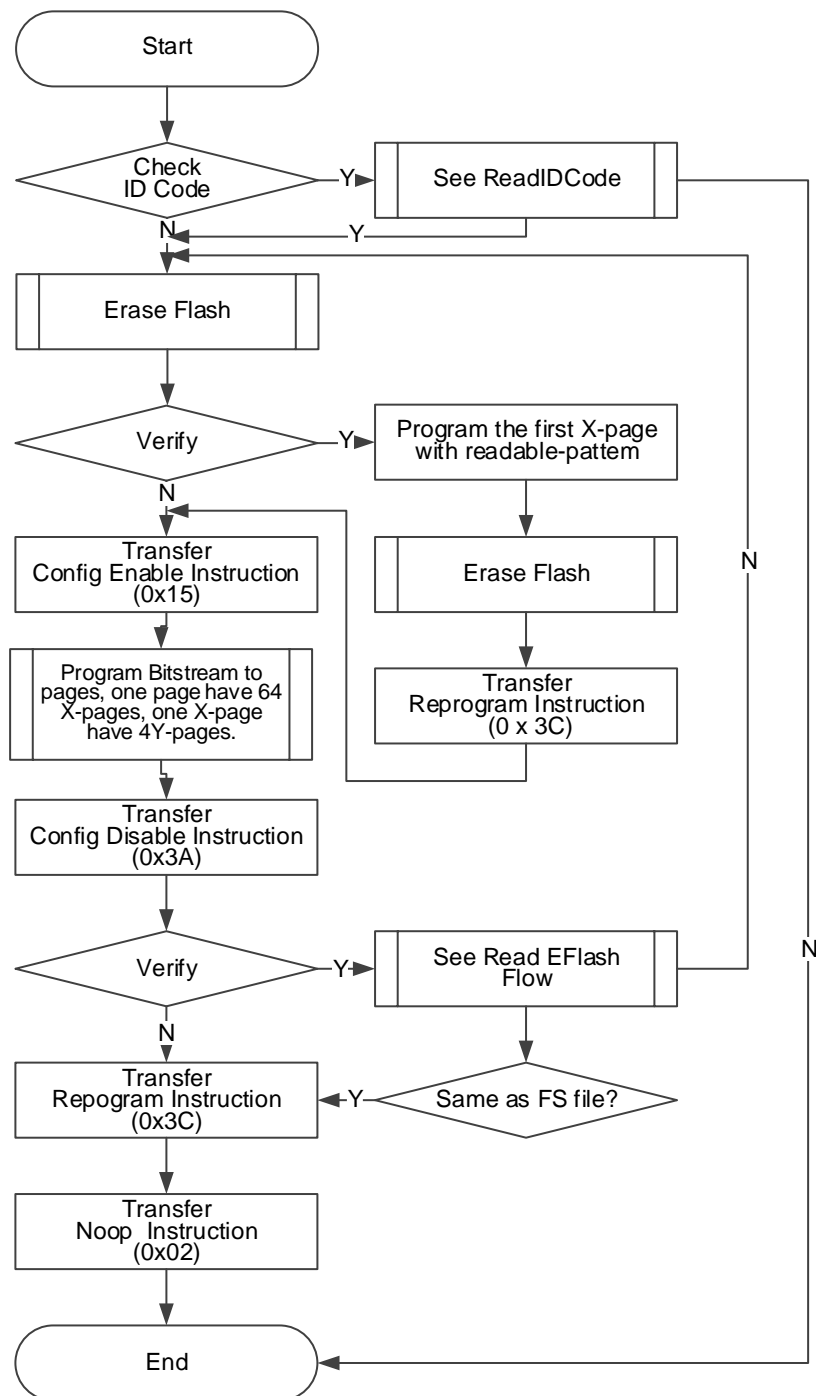
The requirements for JTAG programming frequency are different according to the different processes of the embedded Flash in GW1N series. Please refer to Process of Erasing SRAM > Table 7-9 TCK Frequency Requirements for JTAG.

**Table 7-10 Readback-pattern / Autoboot-pattern**

Device	Readable-pattern(4 Bytes)	Autoboot-pattern(4 Bytes)
H process devices	0x07,0x07,0x30,0x40	0x47,0x57,0x31,0x4E
T process devices	0xF7,0xF7,0x3F,0x4F	

The process of programming internal Flash is shown in :

1. Check whether the ID Code matches.
2. Erase the embedded Flash.
3. Verify if the erasure is successful by reading the Status register to check if the device has been restored to the initial state of the die; the background programming devices and the GW1NS series of devices cannot be checked by reading the Status register.
4. Send the "0x15" instruction of ConfigEnable.
5. Write one X-page at a time until the programming is completed.
6. Send the "0x3A" instruction of Config Disable.
7. Send a Reprogram instruction (0x3C) to load the Flash data into the SRAM.
8. Read Status Code/User Code to check if the loading is successful.

**Figure 7-19 Process of Programming Internal Flash View****Process of Programming an X-page**

The process of programming an X-page is as shown in Figure 7-20.

1. Send the "0x15" instruction of ConfigEnable.
2. Send the "0x71" instruction of EF-Program.
3. Enter into Shift-DR and send address data<sup>1</sup>.
4. Write an X-page data.

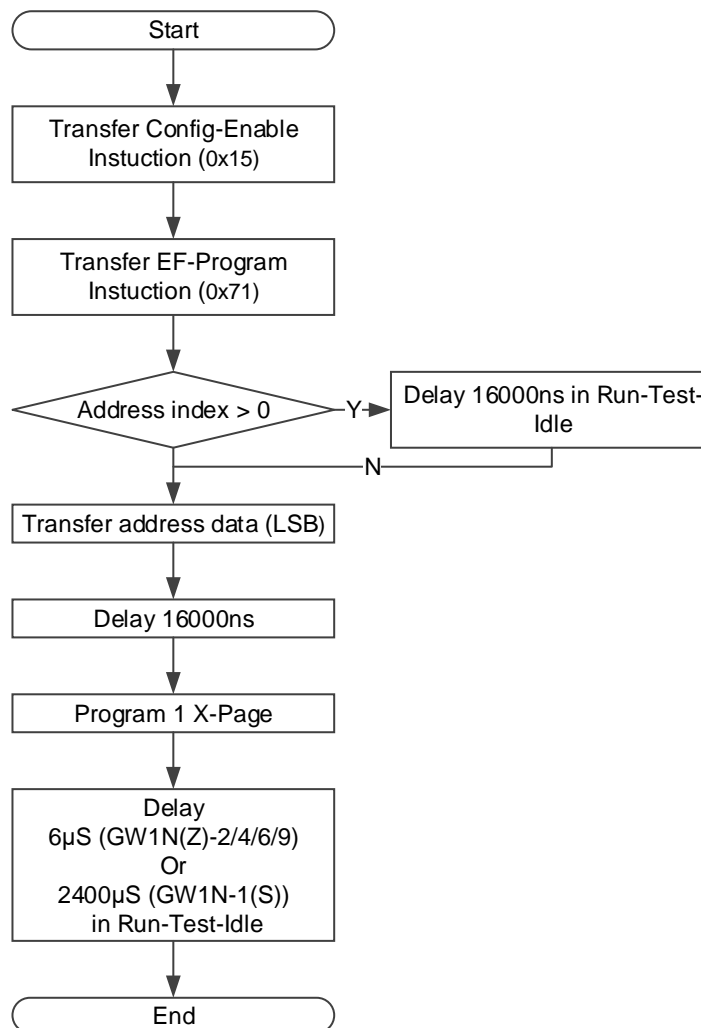
One X-page has 256 bytes in all. Program 4 Bytes and program 64 times for one Y-page. The Y-page data is written in LSB way. Refer to

Figure 7-20.

5. After one X-page is written in, GW1N-1(S) needs to perform Run-Test for 2400 $\mu$ s. GW1N(Z)-2/4/6/9 needs to perform Run-Test for 6 $\mu$ s. No extra clock is required for the other series of devices.
6. This X-page programming ends.

**Note!**

Address data format is 32 bits altogether, and the lower 6 bits are reserved. For example, when the address is b'00010011 (0x13), the written-in address is **000000000000000000000000100110000000**. The address data is written in LSB way. Jump out of Shift-DR at the last bit.

**Figure 7-20 X-page Programming****Process of Programming an Y-page**

Y-page programming is the smallest unit in programming process. 4 Bytes are written each time in the LSB way, as shown in Figure 7-21.

Different series of devices all need to perform Run-Test to wait for writing all Bytes, and the JTAG clock needs to meet minimum frequency requirements. Refer to Table 7-9.

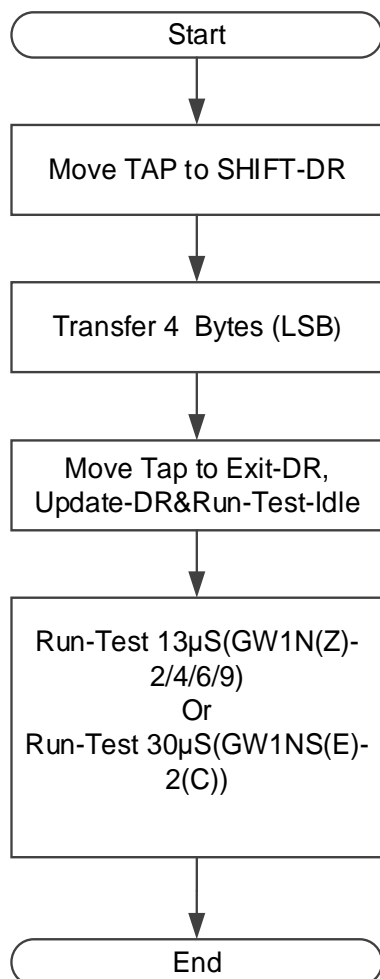
After one Y-page is written in each time, GW1N(Z)-2/4/6/9 needs to perform Run-Test for 13-15  $\mu$ s; GW1N-2(C) needs to perform Run-Test for

30-35  $\mu$ s. No extra clock is required for the other series of devices.

**Note!**

If you want to read data from Configuration Data, high 4 Bytes will be taken. If you want to write data into Shift-DR, LSB will begin to write.

**Figure 7-21 Y-page Programming**



**Process of Reading internal Flash**

This chapter introduces the process of reading internal Flash briefly, no rate requirements for the TCK of JTAG, as shown in Figure 7-22.

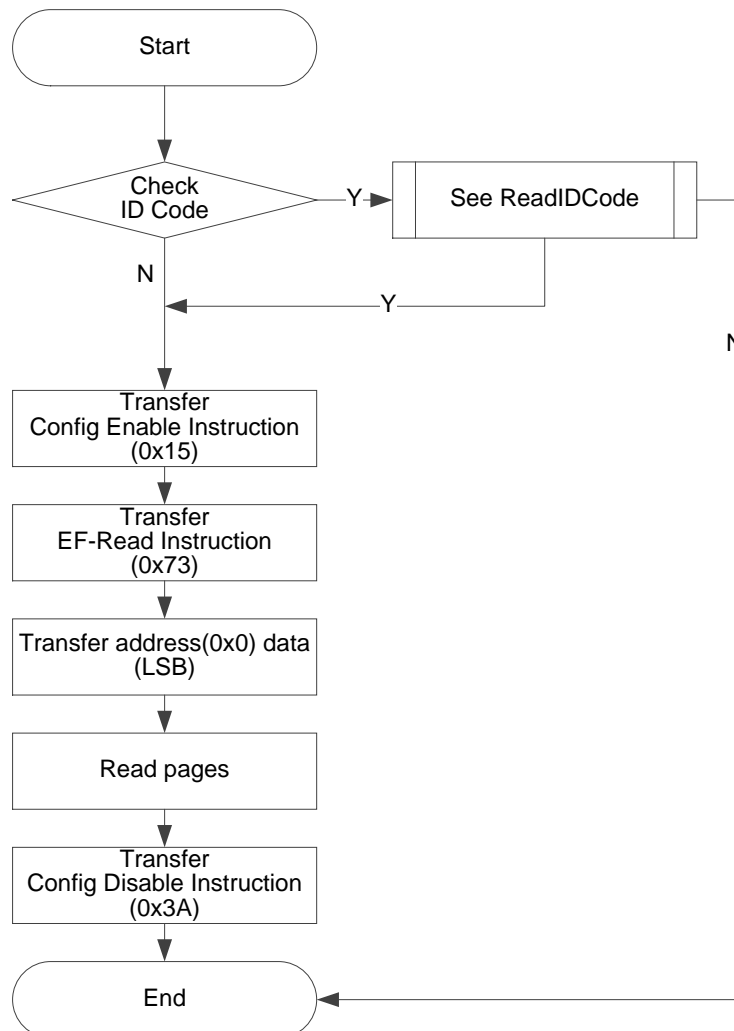
Reading the internal Flash can be regarded as the reverse process of programming Flash. But firstly, you should make sure that the written-in Readable-pattern has taken effect. For GW1N, the Reprogram(0x3C) and Noop(0x02) can be sent in turn after Readable-pattern is written-in to make the internal flash be Readable.

Process Description:

1. Check ID Code. (optional).
2. Send the "0x15" instruction of ConfigEnable.
3. Send EF-Read instruction 0x73.
4. Send read Flash start address 0x0. The method is same as write X-address in Process of Programming Internal Flash.
5. 64 Y-page is an X-page.

6. After reading one X-page, need not to send address again. The address will recurse automatically.
7. After reading, send the "0x3A" instruction of ConfigDisable to end the process.

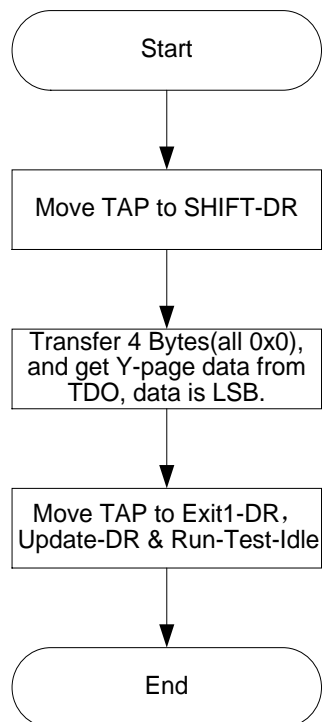
**Figure 7-22 Process of Reading Internal Flash**



#### **Process of Reading a Page (Y-page) Flash**

Reading a Y-page is similar to writing a Y-page, but there is no waiting time for writing in Flash. As shown in Figure 7-23.

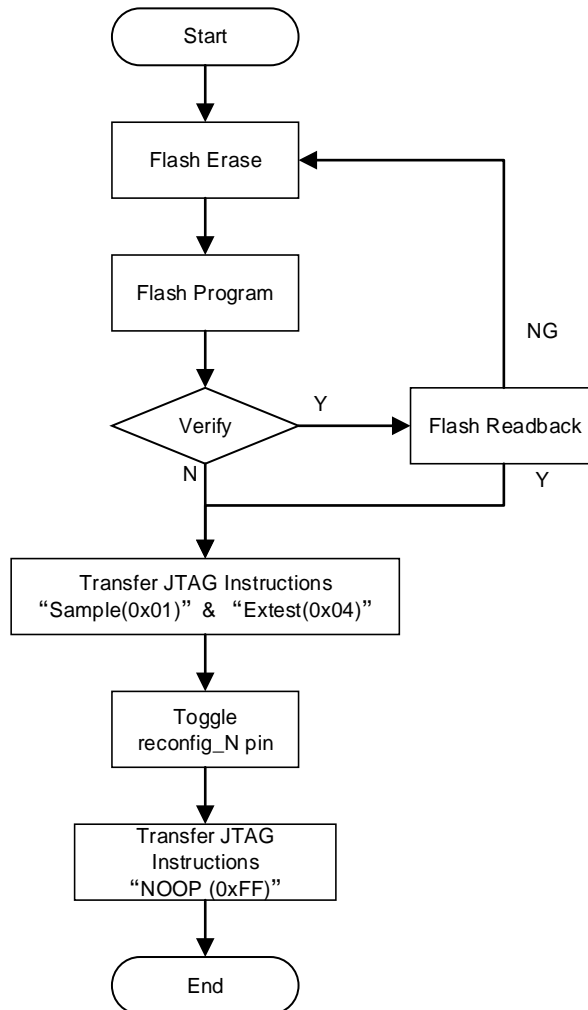
The lowest bit in the data is outputted first.

**Figure 7-23 Process of Reading a Y-page**

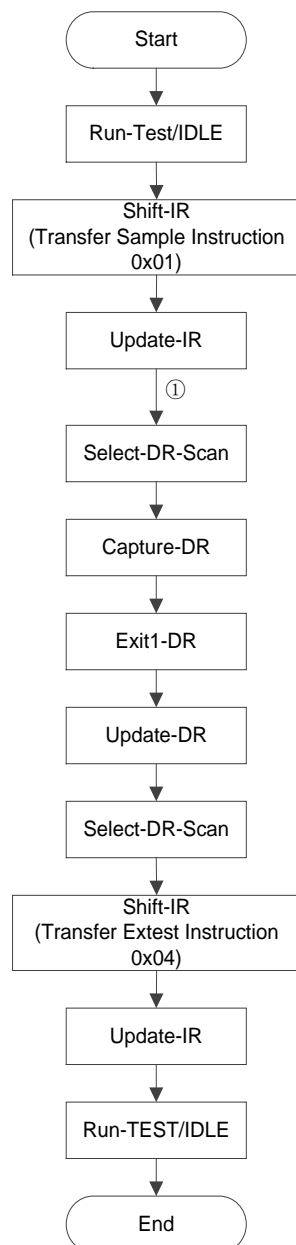
## Background Programming

The device sometimes needs to upgrade the data file and program the Flash without affecting current functions. And it can maintain the I/O state when adding a new data stream file. The following is the flow of GW1N-4 that upgrades the internal Flash data using the Background Programming.

**Figure 7-24 GW1N-4 Background Programming Flow**



The Transfer JTAG Instruction Sample & Extest Flow Chart is as shown in Figure 7-25.

**Figure 7-25 Transfer JTAG Instruction Sample & Exttest Flow Chart****Note!**

① Jump directly from Update-IR to Select-DR-Scan.

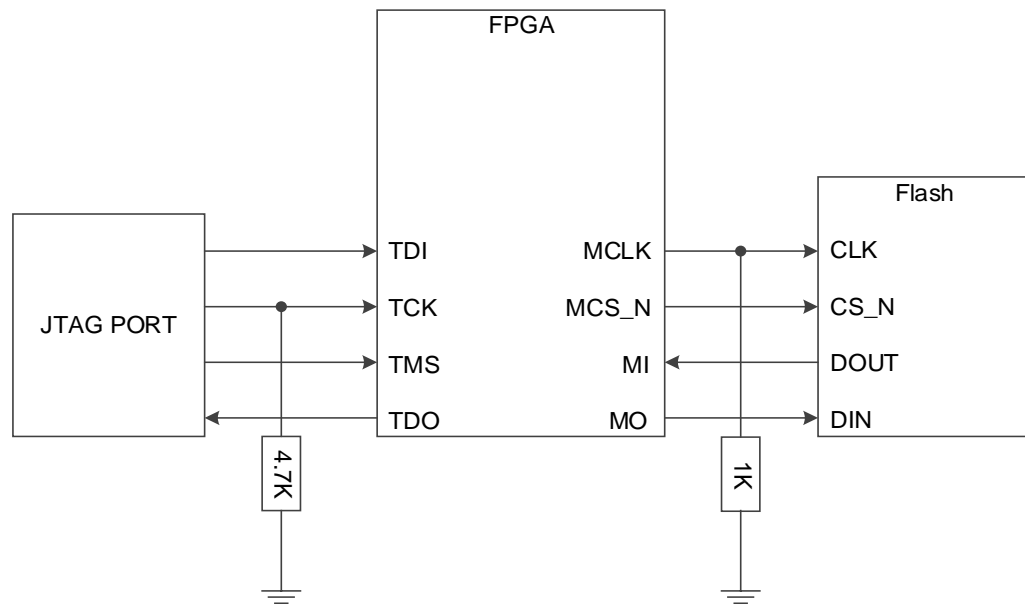
**Programming External Flash or Embedded SPI-Flash**

Gowin FPGAs support loading bitstream files from the external Flash. You can program the external Flash via the JTAG interface directly.

**Note!**

The GW2AN-55 has an embedded SPI Flash, which is programmed in the same way as the GW2A-18 and GW2A-55. The four external pins(MCLK, MCS\_N, MI, and MO) of the GW2AN-55 must be left floating.

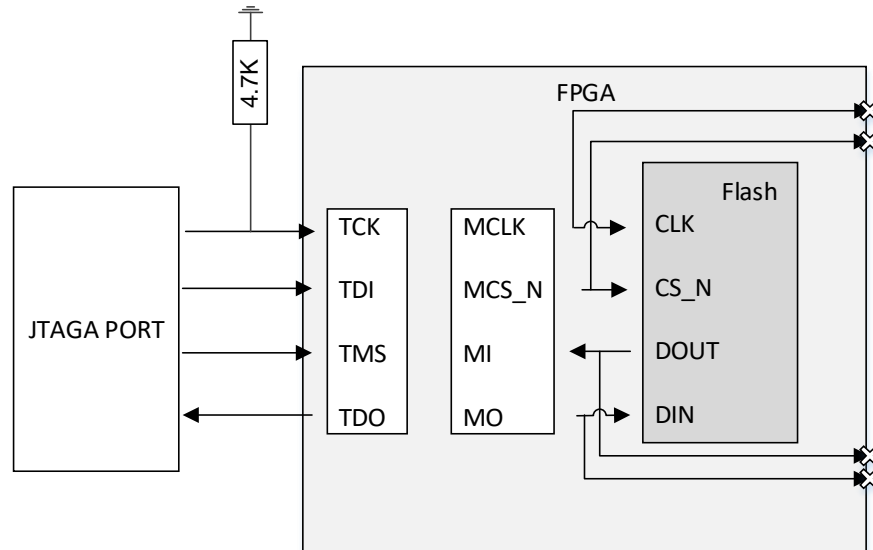
**Figure 7-26 Connection Diagram of Programming External Flash via JTAG Interface (GW2A(R)-18/GW2A-55/LittleBee® Family)**



**Note!**

The figure above shows the minimum system diagram of programming external Flash via the JTAG interface.

**Figure 7-27 Connection Diagram of Programming Embedded SPI Flash via JTAG Interface (GW2AN-55)**



**Note!**

The figure above shows the minimum system diagram of programming embedded SPI Flash via the JTAG interface. The four external pins for MSPI mode must be left floating.

**Programming External Flash via JTAG-SPI**

In this mode, the external Flash can be programmed via the JTAG interface.

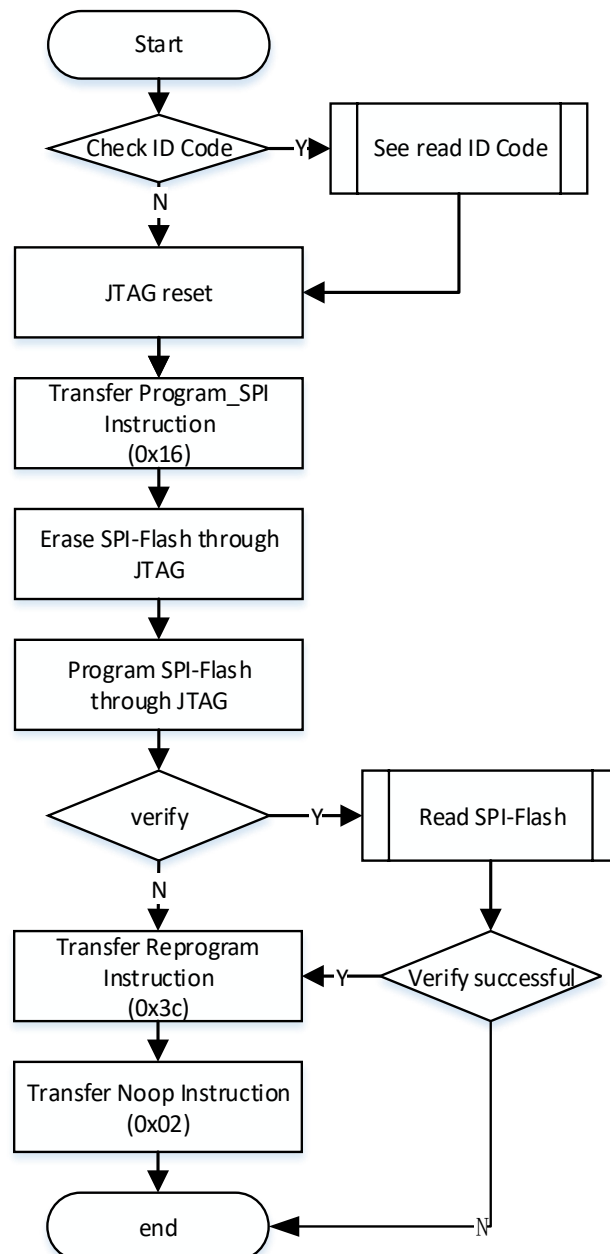
The principle of this mode is to convert the JTAG protocol to the SPI protocol in the form of signal transferring. In this mode, you can program

the SPI Flash by emulating the Master SPI timings via the JTAG interface.

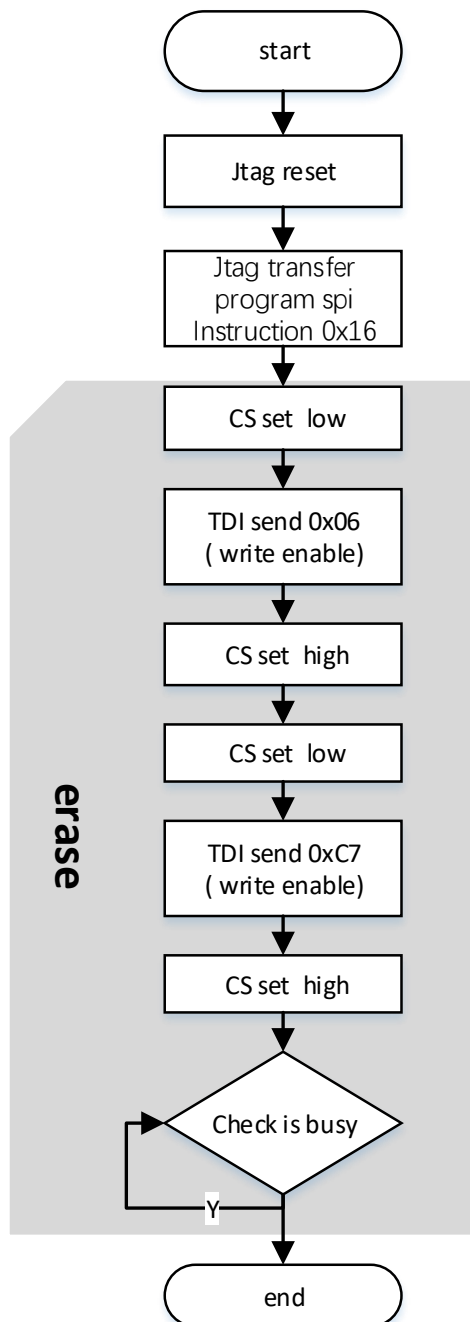
**Note!**

- After sending 0x16, the FPGA transfers the JTAG signal to the MSPI pins to configure the SPI Flash. This transferring function will be disabled when the JTAG is reset.
- When reading back data from the SPI Flash, the data of the first clock cycle is invalid. For example, when reading back the Flash ID code, after sending the 0x9F instruction, an additional clock is needed before reading back 3 bytes of data.
- The JTAG needs to emulate the SPI timings in the SHIFT-DR state.

**Figure 7-28 Process of Programming SPI Flash**



The process of erasing the SPI Flash is as shown in Figure 7-29.

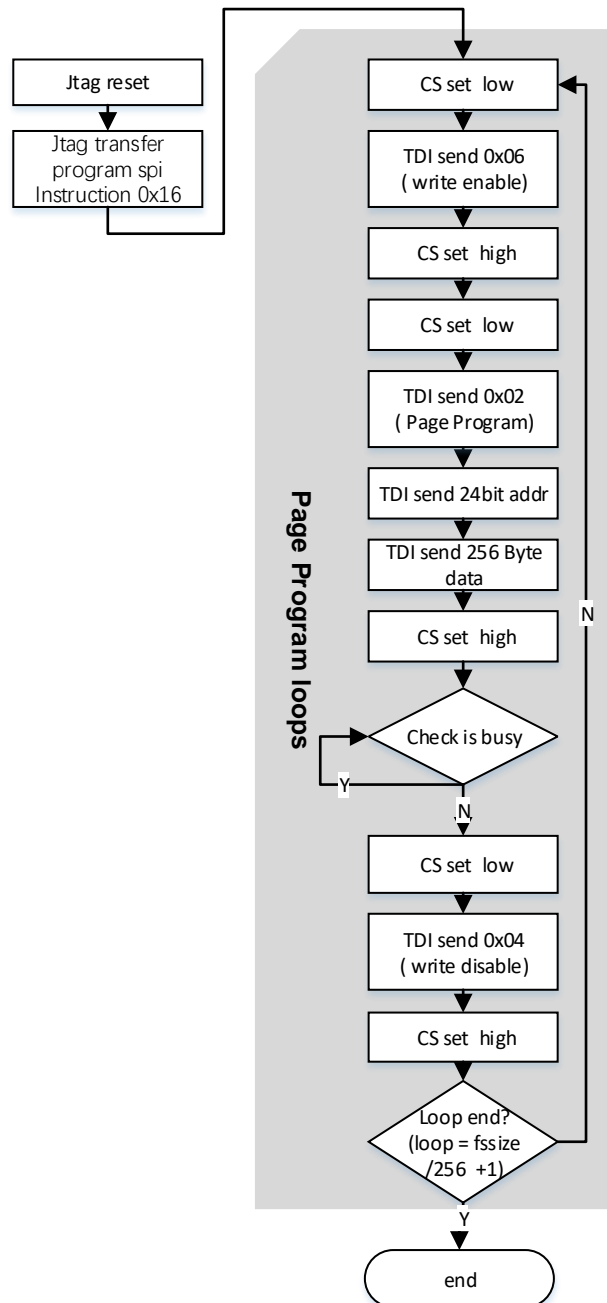
**Figure 7-29 Process of Erasing SPI Flash**

The process of erasing the SPI Flash:

1. JTAG reset.
2. JTAG transfers program SPI instruction 0x16 (LSB).
3. JTAG signals(TCK, TMS, TDI, and TDO) connect to MCLK, CS, MOSI, MISO respectively.
4. Pull CS low and make MOSI write instruction 0x06 with JTAG.
5. Pull CS high with JTAG.
6. Pull CS low and make MOSI write instruction 0xC7 with JTAG.
7. Pull CS high with JTAG.
8. Check if SPI is busy.
9. End of erasure.

The process of programming a page of the SPI Flash is shown in Figure 7-30. The SPI Flash is programmed on a page-by-page basis in a loop.

**Figure 7-30 Process of Programming a Page of the SPI Flash**



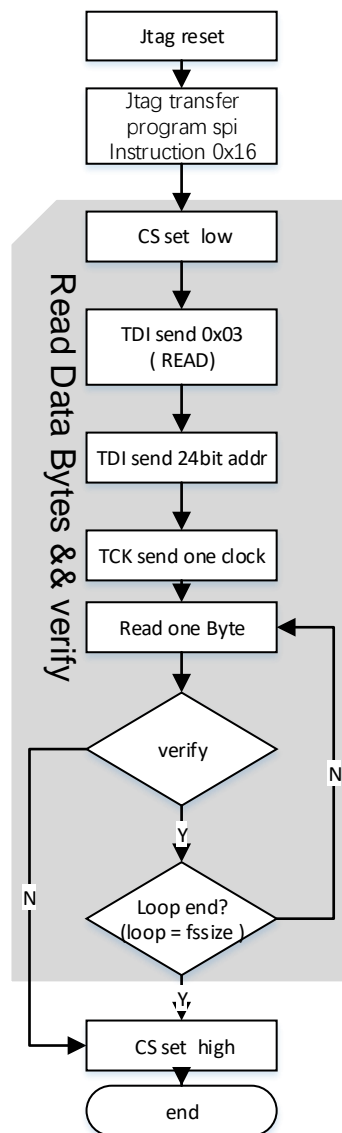
The process of programming a page of the SPI Flash:

1. JTAG reset.
2. JTAG transfers program SPI instruction 0x16 (LSB).
3. JTAG signals(TCK, TMS, TDI, and TDO) connect to MCLK, CS, MOSI, MISO respectively.
4. Pull CS low and make MOSI write instruction 0x06 with JTAG.

5. Pull CS high with JTAG.
6. Pull CS low and make MOSI write instruction 0x02, 3-byte address, and 256-byte fs data with JTAG.
7. Pull CS high with JTAG.
8. Check if SPI is busy.
9. Pull CS low and make MOSI write instruction 0x04 with JTAG.
10. Pull CS high with JTAG.
11. End of programming a page.

The process of reading back SPI Flash and verifying the data stream file is as shown in Figure 7-31.

**Figure 7-31 Process of Reading Back SPI Flash and Verifying the Data Stream File**



The process of reading back SPI Flash and verifying the data stream file:

1. JTAG reset.
2. JTAG transfers program SPI instruction 0x16 (LSB).
3. JTAG signals(TCK, TMS, TDI, and TDO) connect to MCLK, CS, MOSI, MISO respectively.

4. Pull CS low and make MOSI write instruction 0x03 and 3-byte data with JTAG.
5. Make MCLK send a clock with JTAG.
6. JTAG reads back the data, 1 byte at a time.
7. Compare the read back data with the written data stream file. if the two data are the same, continue to compare the next byte until the last byte; if not, jump out of the loop.
8. Pull CS high with JTAG.
9. End of reading back and verifying the data.

Figure 7-32 Timing diagram of Sending 0x06 by Emulating SPI with JTAG(GW2A)

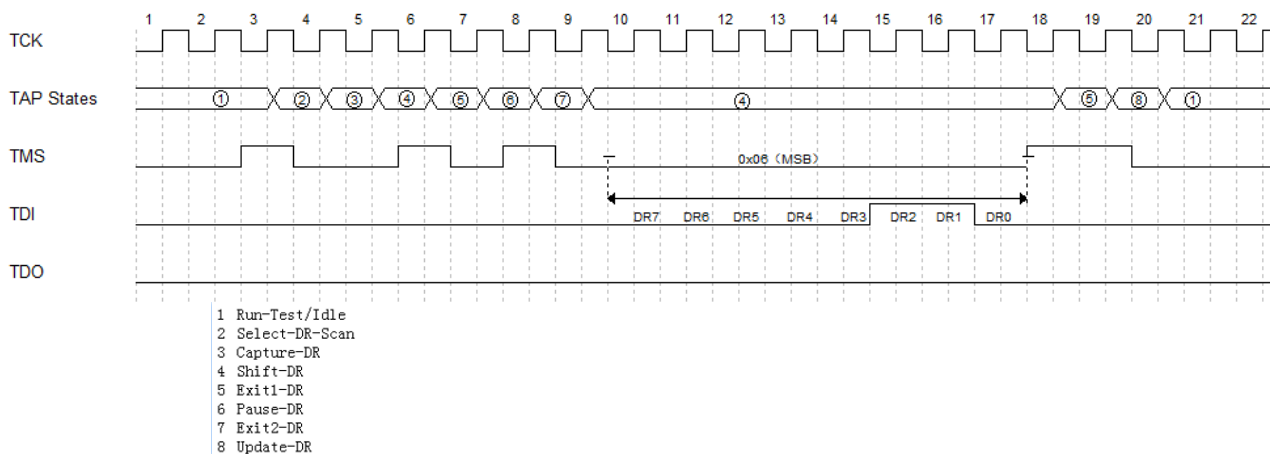
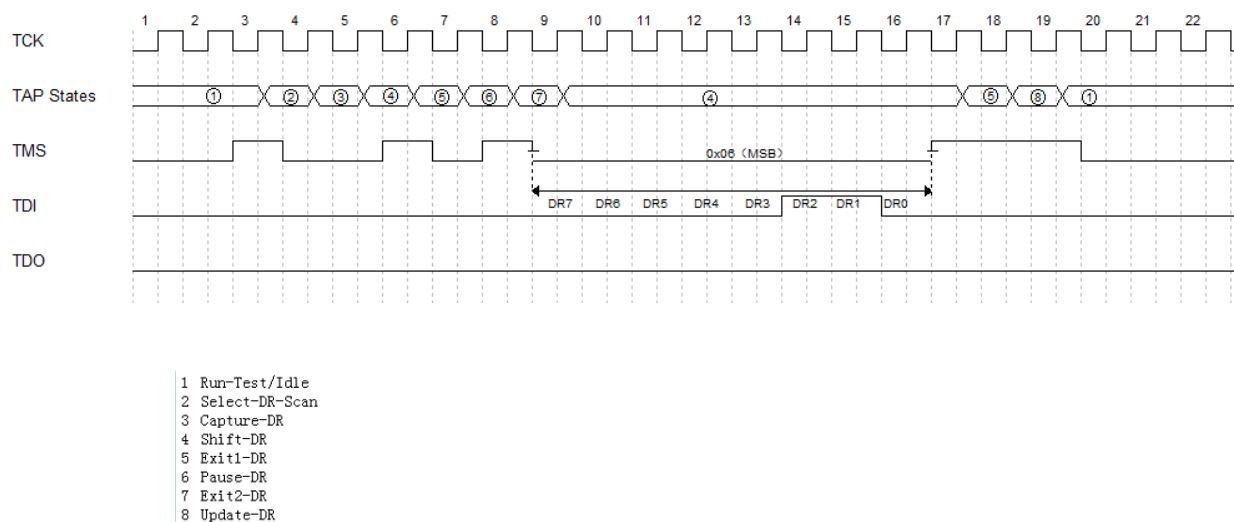


Figure 7-33 Timing diagram of Sending 0x06 by Emulating SPI with JTAG(GW1N)



### Programming SPI Flash in JTAG Boundary Scan Mode

The principle of this mode is changing the state of the pins connected to SPI by using Boundary Scan method to implement SSPI timing, and then to program the internal Flash.

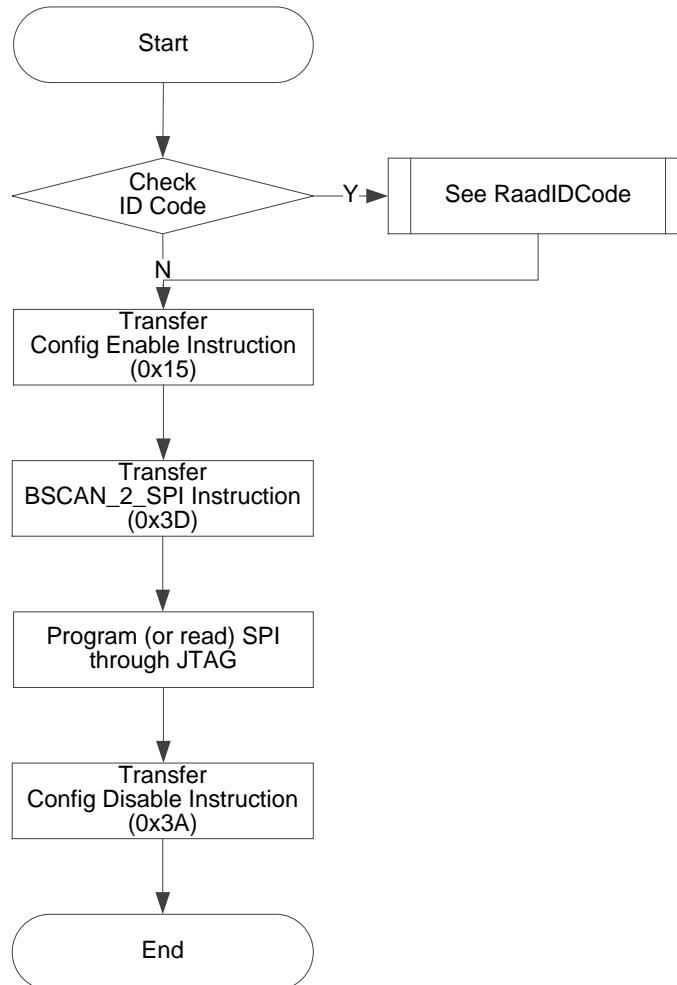
The length of the Boundary Scan Chain used in this mode is 8 bits. Every two bits combination corresponds to the pin state, as shown in Table 7-11. One SCLK drive is completed every two times of sending Boundary Scan Chain.

**Table 7-11 Pin State**

Pins Name of SPI Flash	SCLK		CS		DI		DO	
Bscan Chain[7:0]	7	6	5	4	3	2	1	0
(ctrl & data)	0		0		0		1	

**Note!**

- ctrl:0 means output, 1 means input.
- data:0 means low, 1 means high.

**Figure 7-34 Process of Using Boundary Scan Mode To Program SPI Flash**

### Read Status Register 0x41

The status register can be used to determine if wakeup initialisation and bitstream download or loading were successful or not.

The 32-bit Status Register read instruction is 0x41 and the read timing is the same as the Read ID Code timing.

The Status Register fields are shown in Table 7-12 to Table 7-14.

**Table 7-12 Status Register Definition(I)**

Status Register[31:0]	Device
	GW1N(R)-(1/4B/4C/4D)/GW1NRF-4B
0	CRC Error Flag(Active High)
1	Bad Command Error Flag(Active High)
2	Verify IDCODE Failed Flag(Active High)
3	Timeout Error Flag(Active High)
4	0
5	Memory Erase
6	Preamble
7	Edit Mode
8	program SPI directly
9	0
10	Non-jtag active
11	bypass state
12 <sup>[1]</sup>	Gowin VLD(1: Normal, 0: Error)
13	Done Final(1: Loading Successful, 0: Loading Failed)
14	Security Final(1: Security Enabled, 0: Security Disabled)
15	Ready Status Flag(Active High)
16	POR Success Flag(Active High)
17-31	0

**Note!**

[1] Gowin VLD is associated with the embedded Flash.

**Table 7-13 Status Register Definition(II)**

Status Register[31:0] \ Device	GW1N(R)-(1P5/2/6/9/9C)/GW1NS-4(4C)/ GW1NSR-4(4C)/GW1NSE-4C/GW1NSER-4C/GW1NZ-(1/2)
0	CRC Error Flag(Active High)
1	Bad Command Error Flag(Active High)
2	Verify IDCODE Failed Flag(Active High)
3	Timeout Error Flag(Active High)
4	0
5	Memory Erase
6	Preamble
7	Edit Mode
8	program SPI directly
9	AutoBoot State
10	Non-jtag active
11	bypass state
12 <sup>[1]</sup>	Gowin VLD(1: Normal, 0: Error)
13	Done Final(1: Loading Successful, 0: Loading Failed)
14	Security Final(1: Security Enabled, 0: Security Disabled)
15	Ready Status Flag(Active High)
16	POR Success Flag(Active High)
17	Flash Lock
18-31	0

**Note!**

[1] Gowin VLD is associated with the embedded Flash.

**Table 7-14 Status Register Definition(III)**

Status Register[31:0] \ Device	GW2A(NR)-18/18C/55/55C
0	CRC Error Flag(Active High)

Status Register[31:0]	Device
	GW2A(NR)-18/18C/55/55C
1	Bad Command Error Flag(Active High)
2	Verify IDCODE Failed Flag(Active High)
3	Timeout Error Flag(Active High)
4	0
5	Memory Erase
6	Preamble
7	Edit Mode
8	program SPI directly
9	0
10	Non-jtag active
11	bypass state
12	0
13	Done Final(1: Loading Successful, 0: Loading Failed)
14	Security Final(1: Security Enabled, 0: Security Disabled)
15	Encryption Format(1: Encryption Enabled, 0: Encryption Disabled)
16	Encryption Key Match(1: Correct Key, 0: Wrong Key)
17-31	0

### GW1N FPGA Family Device Programming

Status Register Bit-15 READY only returns 0x0 if something goes wrong during programming. E.g., CRC Error, Bad-command, IDCODE mismatch etc.

If Status Register Bit-15 READY returns 0x0, check Status Register Bits[3:0] to determine the cause of the Download Error.

Status Register Bit-13 DONE should NOT be used standalone to check if the download was successful. DONE MUST always be checked in conjunction with READY (see above).

### GW1N FPGA Family Devices Status Register Return Values

0x0001B020 (Security bit NOT set) means the FPGA has been successfully configured (this is not recommended for production, as download data can be read from SRAM).

0x0001F020 (Security bit set) also means the FPGA has been successfully configured.

I.e. Successful Download status register return value:

- Bit-16 POR Status = 0x1
- Bit-15 Ready Status = 0x1
- Bit-14 Security Status = 0x1 or 0x0 (see above)
- Bit-13 DONE Final Status = 0x1
- Bit-12 VLD = 0x1

#### **GW2A FPGA Family Device Programming**

When Programming GW2A devices, the following bits are only used during the programming sequence, and once programmed these bits are automatically cleared. I.e. the final status return value for these two bits will always be 0x0.

- Bit-15 Encryption Format
- Bit-16 Encryption Key Match

In addition, the GOWIN VLD status bit is only applicable to devices that have embedded Flash. Therefore, GW2A Bit-12 will also return 0x0.

#### **GW2A FPGA Family Devices Status Register Return Values**

0x02020 (Security bit NOT set) means the FPGA has been successfully configured (this is not recommended for production, as download data can be read from SRAM)

0x06020 (Security bit set) means the FPGA has been successfully configured.

I.e. Successful Download status register return value:

- Bit-16 = 0x0
- Bit-15 = 0x0
- Bit-14 Security Status = 0x1 or 0x0 (see above)
- Bit-13 DONE Final Status = 0x1
- Bit-12 = 0x0

For more information on the Status Register, please refer to [TN711, GOWIN FPGA Status Register Codes](#).

#### **Read Code 0x13**

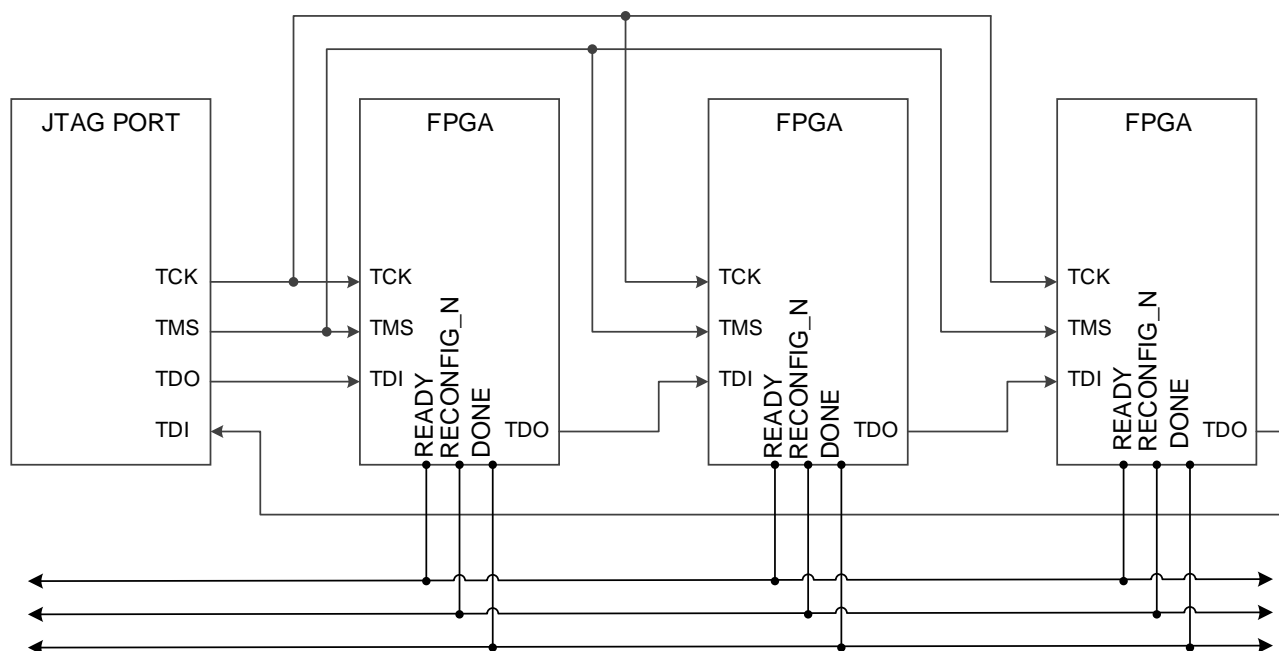
The user code is 32 bits, the read instruction is 0x13 and the timing is the same as that of Read ID Code.

The user code adopts the checksum value in the FS file by default. It can be redefined using Gowin Designer.

#### **Reload 0x3C**

This instruction is used to read the bitstream files from Flash and write to SRAM.

Send the instructions of Reprogram (0x3C) and Noop (0x02) to reload the device via JTAG. You can also reload the device by triggering the Reconfig\_N pin.

**Connection Diagram of Daisy-Chain****Figure 7-35 Connection Diagram of Daisy-Chain****Routine File**

For the routine file, please contact GOWINSEMI technical support or the local office.

## 7.3 AUTO BOOT Configuration Mode (Supported by LittleBee® Family Only)

The AUTO BOOT mode is a configuration mode for momentary connection feature of non-volatile LittleBee® family of FPGA Products. The Arora Family of FPGA products do not support AUTO BOOT mode. In AUTO BOOT mode, FPGA reads bitstream data from the built-in Flash automatically after it is powered on, with no connection to an external configuration interface.

In the AUTO BOOT mode, the bitstream data needs to be written to the built-in Flash via the JTAG port first (refer to Figure 7-4 Connection Diagram for JTAG Configuration Mode), and then set the MODE value to "000", the chip will automatically read the bitstream data to complete configuration when powered up again or RECONFIG\_N triggered at a low-level pulse. When the MODE value is set to "000", the FPGA will automatically configure the SRAM to complete AUTO BOOT after the built-in Flash is programmed using Gowin programmer. The momentary connection feature of the built-in Flash saves download time and improves productivity.

GW1N(R) - 9 and GW1NS series support two retries of AUTO BOOT configuration, i.e. the devices can be automatically reconfigured if the first configuration fails after power up. The other devices of LittleBee® only support one-time AUTO BOOT configuration. The factors that can lead to a

failed configuration include ID validation error, CRC check error, and instruction error.

**Note!**

The embedded Flash can only store one bitstream file. The retry address configuration could not be changed

## 7.4 SSPI Configuration Mode

In SSPI (Slave SPI) mode, FPGA is as a slave device and is configured via SPI by an external Host.

### 7.4.1 SSPI Mode Pins

The SSPI configuration pins are shown in Table 7-15.

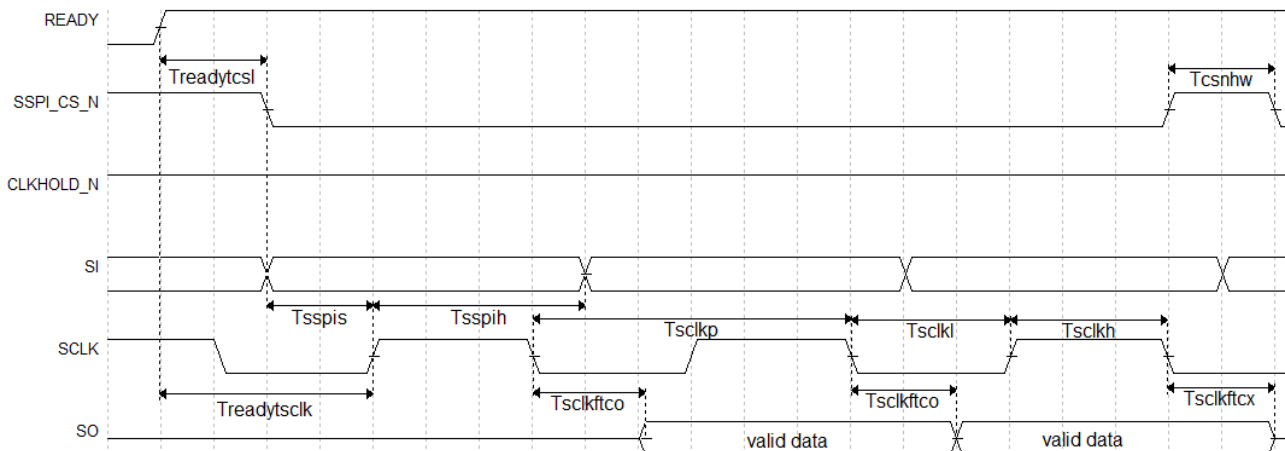
Table 7-15 SSPI Mode Pins

Pin Name	I/O	Description
RECONFIG_N	I, Internal weak pull- up	Low level pulse: Start GowinCONFIG
READY	I/O	High level: FPGA can be programmed and configured. Low level: Programming configuration for FPGA is prohibited.
DONE	I/O	High-level pulse: Successfully programmed and configured. Low-level pulse: Programming and configuration uncompleted or failed.
MODE[2:0]	I, Internal weak pull- up	Configuration mode selection, READY rising edge sampling
SCLK	I	Input clock
CLKHOLD_N	I, Internal weak pull- up	Active high
SO	O	FPGA outputs data to Host
SI	I	Input data to FPGA from Host
SSPI_CS_N	I, Internal weak pull- up	SSPI Chip selection signal, active low.

### 7.4.2 SSPI Configuration Timing

See Figure 7-36 for the SSPI timing.

### Figure 7-36 SSPI Configuration Timing



See Table 7-16 for the SSPI configuration timing parameters.

### Table 7-16 SSPI Configuration Timing Parameters

Name	Description	Min.	Max.
T <sub>sclkp</sub>	SCLK clock period	15ns	-
T <sub>sclkh</sub>	SCLK clock high time	7.5ns	-
T <sub>sclkl</sub>	SCLK clock low time	7.5ns	-
T <sub>sspis</sub>	SSPI PORT setup time	2ns	-
T <sub>sspih</sub>	SSPI PORT hold time	0ns	-
T <sub>sclktco</sub>	Time from SCLK falling edge to output	-	10ns
T <sub>sclktcx</sub>	Time from SCLK falling edge to high impedance	-	10ns
T <sub>csnhw</sub>	CSN high time	25ns	-
T <sub>readytcsi</sub>	Time from READY rising edge to CSN low	10μs	
T <sub>readytsclk</sub>	Time from READY rising edge to first SCLK edge	10μs	-

Other than the power requirements, the following conditions need to be met to use the SSPI configuration mode:

- SSPI port enable  
RECONFIG\_N is not set as a GPIO during the first configuration after power up or the previous programming.
- Initiate new configuration  
Power up again or trigger RECONFIG\_N at one low pulse.

### 7.4.3 Configuration Instruction

In Slave SPI mode, you can program FPGA SRAM or read ID information on ID CODE\USER CODE\STATUS CODE through SSPI.. External memory can also be programmed (Such as SPI Flash).

The SSPI instruction of FPGA is generally composed of 1-4 bytes, including at least 1 instruction class byte and multiple redundant

information bytes. If there is no specified information byte, the redundant information byte can be any number (0x00 is used in the following table).

**Table 7-17 Configuration Instruction**

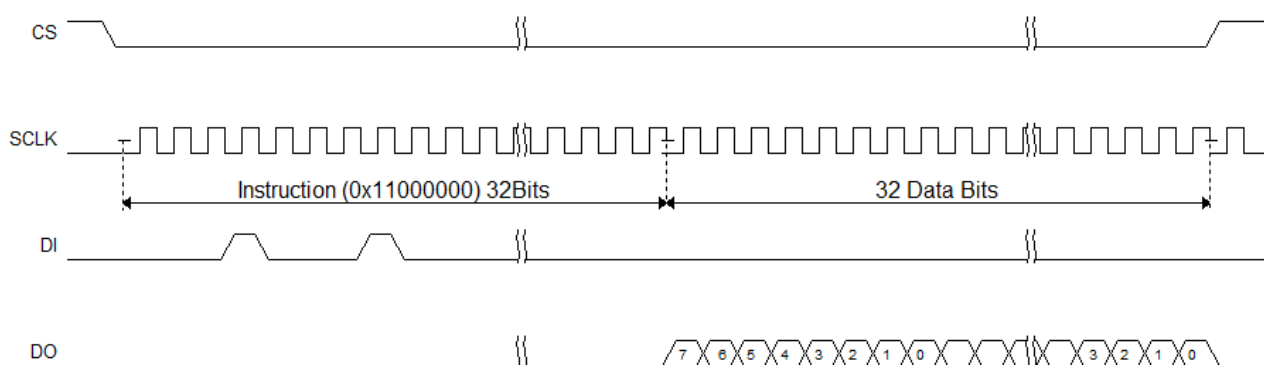
Name	Complete Instruction (Instruction Byte + Redundant Information Byte)
Read ID Code	0x11000000
Read User Code	0x13000000
Read Status Code	0x41000000
Reconfig/Reprogram	0x3C00
Write Enable	0x1500
Write Disable	0x3A00
Write Data	0x3B
Program SPI Flash	0x1600
Init Address	0x1200
Erase SRAM	0x0500

### Read ID Code

The length of FPGA ID Code is 32bits. The instruction to read ID is four bytes, that is 0x11000000. Before sending instructions, keep CS at a high level and generate multiple clocks (more than two) to let FPGA get CS state.

After CS is pulled down, the instruction of 0x11000000 is written in MSB way and after this, 32 clocks are generated continuously. At this time, the ID CODE data will be successively shifted out of DO in the form of MSB.

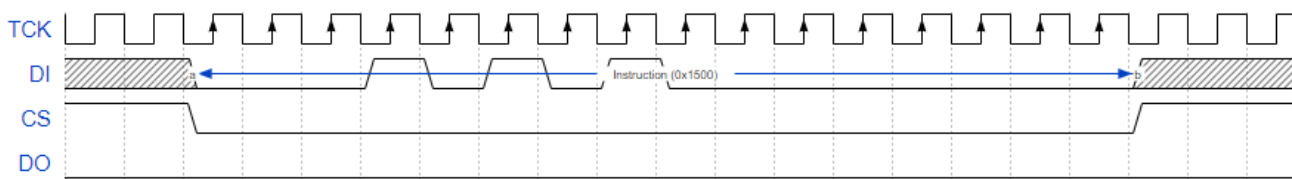
**Figure 7-37 Read ID Code Timing**



The operation of reading Status Code / User Code is similar to the operation of reading ID Code, just replace the corresponding instructions.

### Write Enable (0x1500)

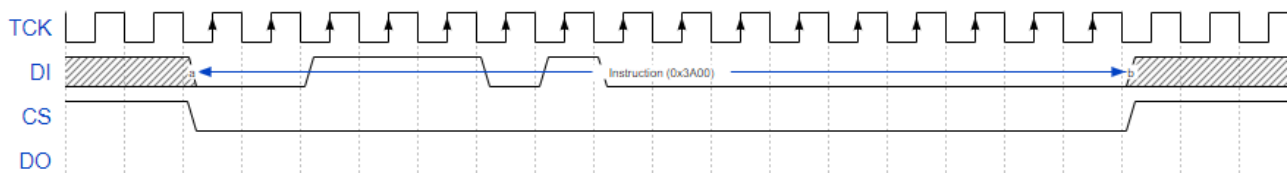
Before configuring SRAM (writing features), enter programming mode using "Write Enable (0x15)" instruction to receive the "Write Data (0x3B)" instructions.

**Figure 7-38 Write Enable (0x15) Timing****Note!**

At CS high level, more than two clocks should be given to SCLK to drive FPGA to identify CS signal. This rule also applies to other instructions.

**Write Disable (0x3A00)**

After finishing sending data, exit programming mode using Write Disable. After exiting, the device can be awakened to enter the working state.

**Figure 7-39 Write Disable(0x3A00) Timing**

The timing of 0x1500 and 0x3A is basically the same. Instructions start at CS low level and the CS is pulled up after the instruction transmission is completed. Instructions following this timing are as follows: 0x3C00 (Reconfig / Reprogram), 0x1500(Write Enable), 0x3A000 (Write Disable), 0x1600(Program SPI Flash), 0x1200(Init Address), 0x0500(Erase SRAM).

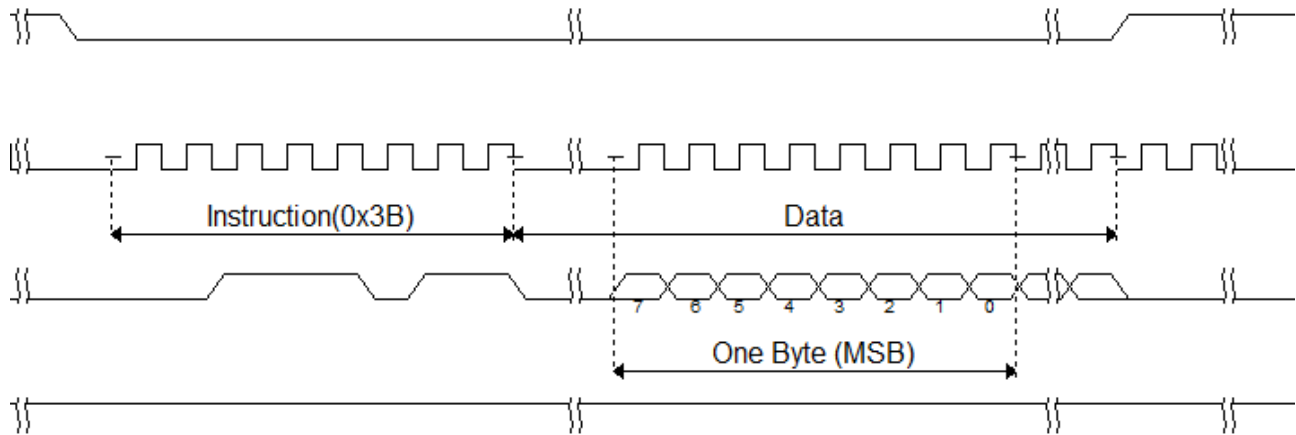
In addition, SSPI is driven by an external clock, so if CS is at high before and after these instructions, more than two clocks are needed to enable FPGA to collect the state of CS.

### Write Data (0x3B)

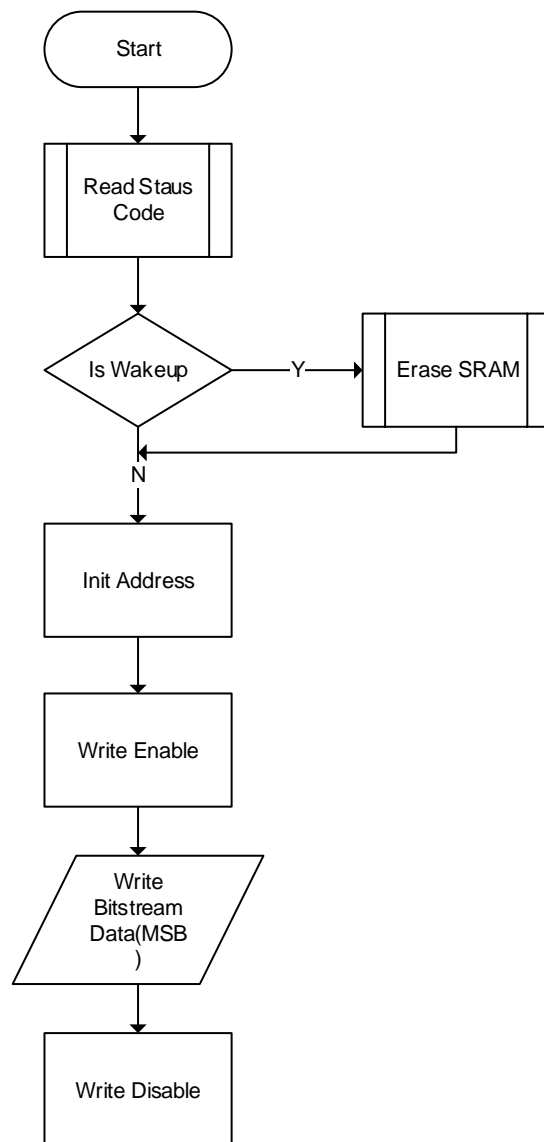
The fs file is sent directly to the FPGA device using the "Write Data (0x3B)" instruction.

Note that CS keeps low level in the process of data writing.

Figure 7-40 Write Data (0x3B) Timing



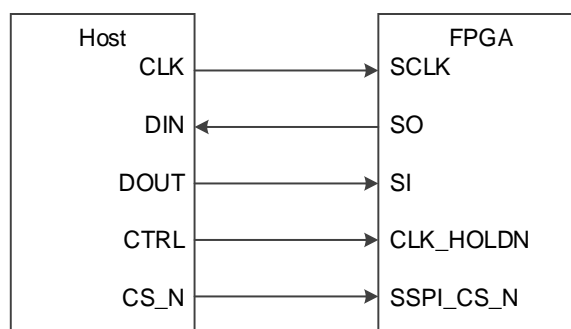
### 7.4.4 The Flow Chart of Configuring SRAM via SSPI



### 7.4.5 Connection Diagram for SSPI Configuration Mode

The connection diagram for configuring Gowin FPGA products via SSPI is shown in Figure 7-41.

**Figure 7-41 SSPI Configuration Mode Connection Diagram**

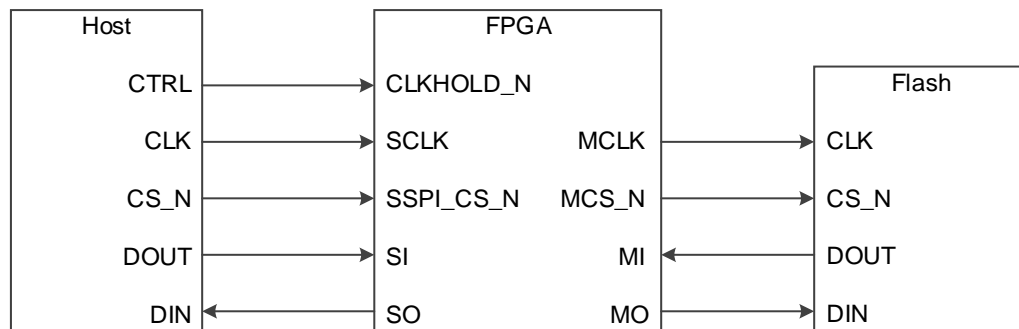


**Note!**

The figure above shows the minimum system diagram for the SSPI configuration. The value of the SSPI MODE is "001". The connection of the other fixed pins is shown in Figure 7-1.

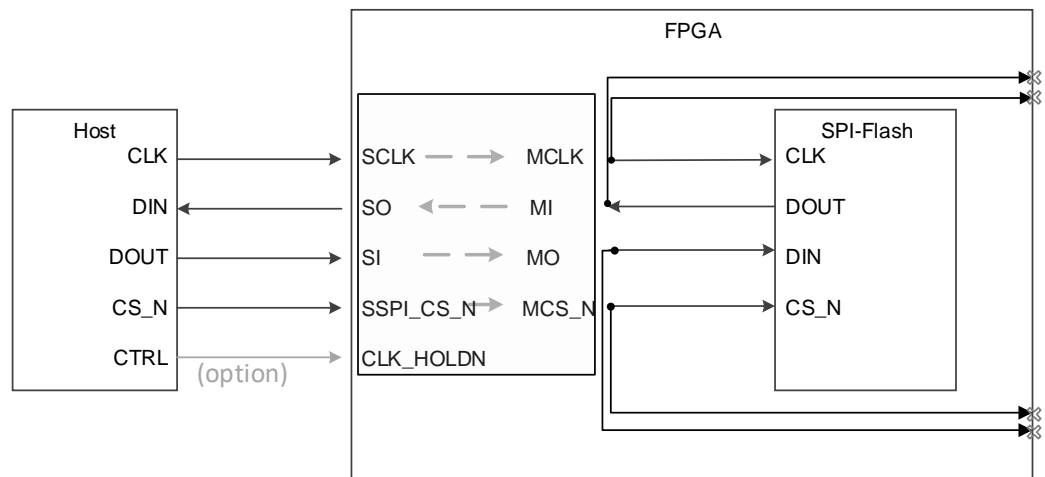
In addition to SRAM, SSPI can be used to program the SPI Flash. The MODE value of the Flash programming is the same as the MODE value of SSPI configuration mode. Configuration data can be written to SRAM or Flash using Gowin programmer. The connection diagrams for programming an external Flash and internal Flash via SSPI are shown in Figure 7-42 and Figure 7-43.

**Figure 7-42 Connection Diagram of Programming External Flash via SSPI(GW2A-18/55,GW1N(R)-9)**

**Note!**

- All Arora family devices support programming external Flash via SSPI.
- For the LittleBee® family devices, currently only GW1N(R)-9 supports programming external Flash via SSPI.

**Figure 7-43 Connection Diagram of Programming Internal Flash via SSPI (GW2AN-55)**

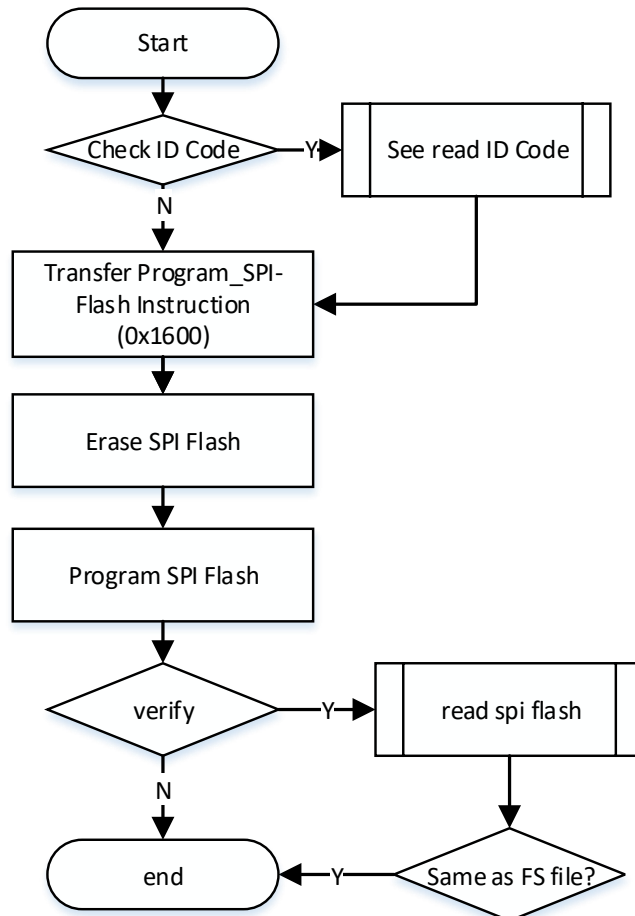
**Note !**

The GW2AN-55 has an embedded SPI Flash, which is programmed in the same way as the GW2A-18 and GW2A-55. The four external pins(MCLK, MCS\_N, MI, and MO) of the GW2AN-55 must be left floating.

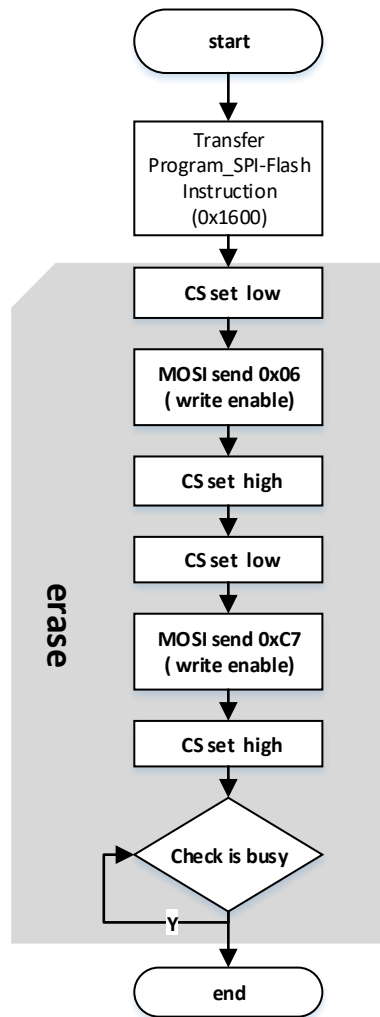
Please refer to Figure 7-44 for the flow of programming SPI Flash via SSPI. First, send the "Program SPI Flash" (0x1600) instruction to the FPGA via SSPI. After this, the FPGA can transfer SSPI to the Flash, and the SSPI on the Host side can directly access the Flash. Then, it can be programmed according to the Flash timing.

Note that when reading data from Flash, the data read back is delayed by one Bit.. For example, when SSPI reads Flash's ID Code, it needs to send an extra Clock to get the last bit.

**Figure 7-44 The Flow Chart of Programming Flash via SSPI**



The process of erasing the SPI Flash is as shown in Figure 7-45 :

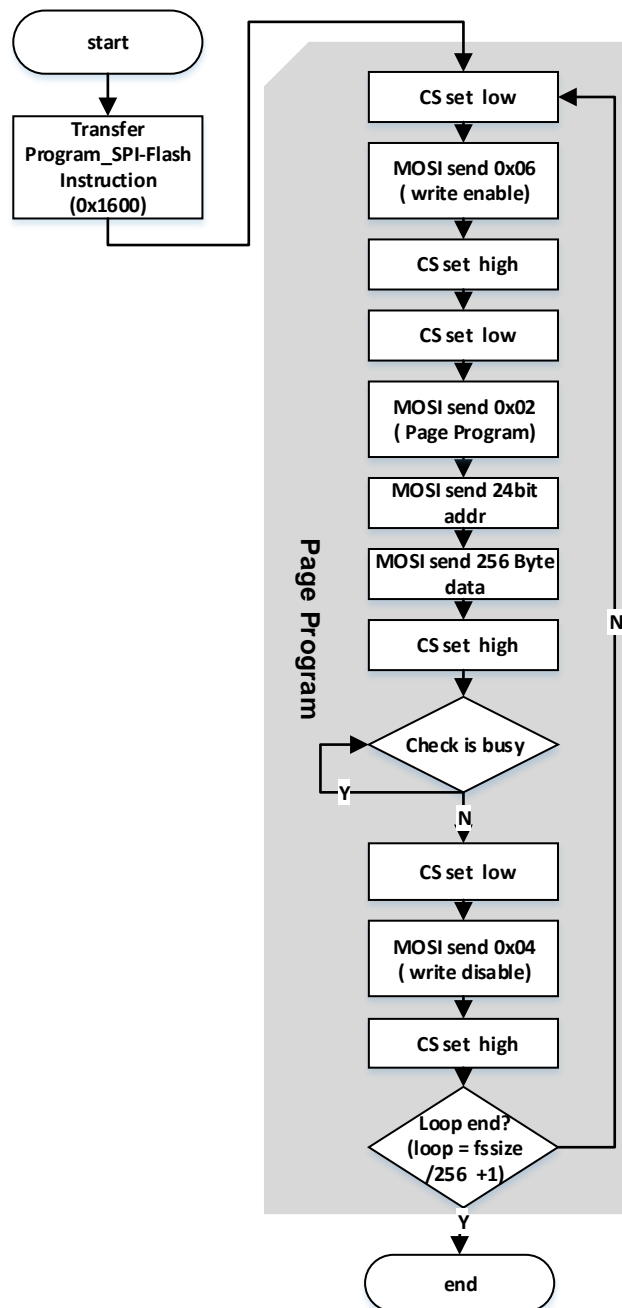
**Figure 7-45 The Flow Chart of Erasing SPI Flash**

The process of erasing the SPI Flash:

1. Start.
2. Host MSPI transfers the program spi instruction 0x1600 (MSB).
3. Device-SSPI signals(SCLK, CS, SI, and SO) connect to MCLK, CS, MOSI, MISO respectively inside the FPGA.
4. Pull CS low and make MOSI write instruction 0x06 with Host MSPI.
5. Pull CS high with Host MSPI.
6. Pull CS low and make MOSI write instruction 0xC7 with Host MSPI.
7. Pull CS high with Host MSPI.
8. Check if the SPI is busy.
9. End of erasure.

The process of programming a page of the SPI Flash is shown in Figure 7-46. The SPI Flash is programmed on a page-by-page basis in a loop.

Figure 7-46 The Flow Chart of Programming a Page of the SPI Flash



The process of programming a page of the SPI Flash:

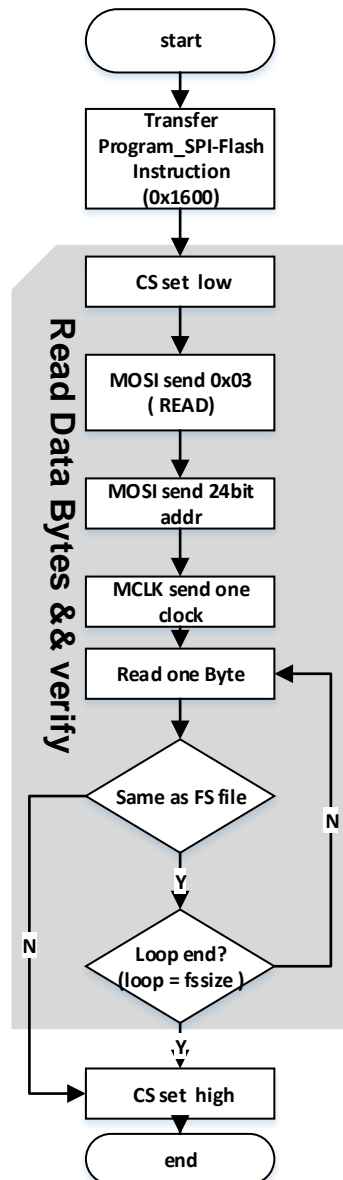
1. Start.
2. Host MSPI transfers the program spi instruction 0x1600 (MSB).
3. Device-SSPI signals(SCLK, CS, SI, and SO) connect to MCLK, CS, MOSI, MISO respectively inside the FPGA.
4. Pull CS low and make MOSI write instruction 0x06 with Host MSPI.
5. Pull CS high with Host MSPI.
6. Pull CS low and make MOSI write instruction 0x02, 3-byte address, and 256-byte fs data with Host MSPI.
7. Pull CS high with Host MSPI.
8. Check if the SPI is busy.
9. Pull CS low and make MOSI write instruction 0x04 with Host MSPI.

10. Pull CS high with Host MSPI.

11. End of programming a page.

The process of reading back SPI Flash and verifying the data stream file is as shown in Figure 7-47:

**Figure 7-47 The Flow Chart of Reading Back SPI Flash and Verifying the Data Stream File**



The process of reading back SPI Flash and verifying the data stream file:

1. Start.
2. Host MSPI transfers the program spi instruction 0x1600 (MSB).
3. Device-SSPI signals(SCLK, CS, SI, and SO) connect to MCLK, CS, MOSI, MISO respectively inside the FPGA.
4. Pull CS low and make MOSI write instruction 0x03 and 3-byte data with Host MSPI.
5. Make MCLK send a clock with Host MSPI.
6. Host MSPI reads back the data, 1 byte at a time.

7. Compare the read back data with the written data stream file. if the two data are the same, continue to compare the next byte until the last byte; if not, jump out of the loop.
8. Pull CS high with Host MSPI.
9. End of reading back and verifying the data.

### 7.4.6 Multiple FPGA Connection View in SSPI Mode

Figure 7-48 Multiple FPGA Connection Diagram 1

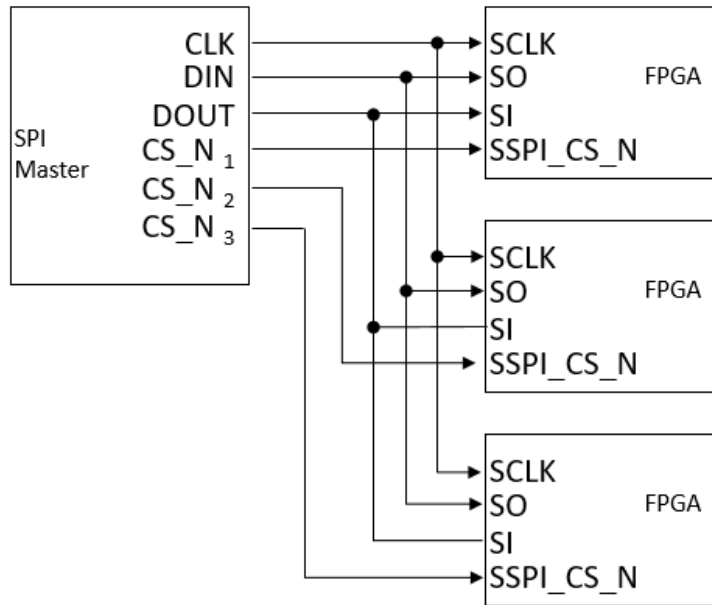
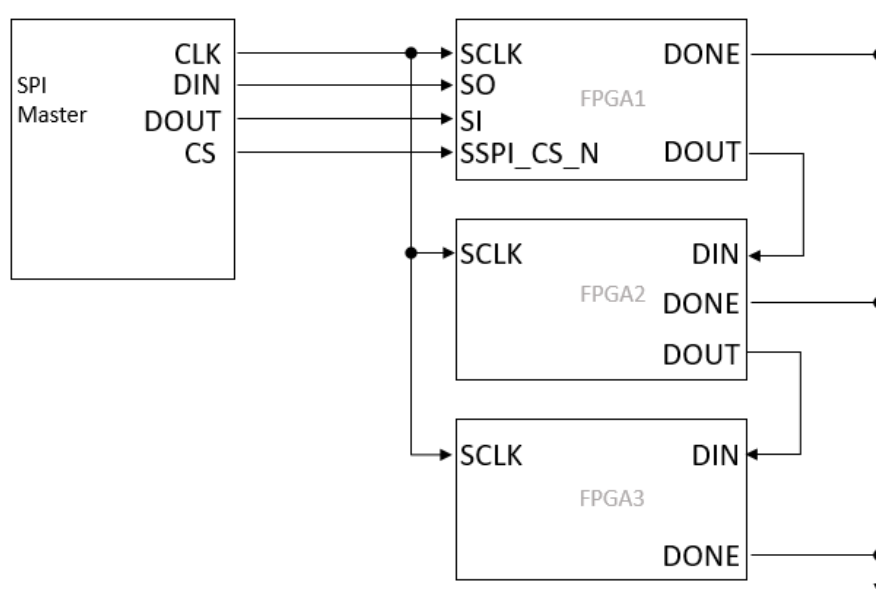


Figure 7-49 Multiple FPGA Connection Diagram 2



## 7.5 MSPI Configuration Mode

In MSPI (Master SPI) mode, the FPGA as the Master reads bitstream data from external Flash memory via its SPI port to configure the FPGA's internal SRAM.

### MSPI Mode FPGA Configuration:

1. Set the MODE pin configuration values to be MSPI mode.
2. To prompt the FPGA to automatically load the bitstream from external Flash either
  - Power Cycle the FPGA
  - Or pulse RECONFIG\_N low.

### MSPI Mode External Flash Update:

The external Flash memory can also be re-programmed via the FPGA using JTAG. This feature enables the FPGA to support bitstream background updates and is often referred to as infield or remote update. Once the FPGA has been configured, users can remotely write new configuration data to the external Flash via the FPGA. Once Flash programming completes the new bitstream can be automatically loaded by triggering RECONFIG\_N or power-cycling the FPGA.

## 7.5.1 MSPI Mode Pins

The configuration of the MSPI mode is shown in Table 6-15.

**Table 7-18 Pin Description in MSPI Configuration Mode**

Pin Name	I/O	Description
RECONFIG_N	I, Internal weak pull-up	Low level pulse: Start Gowin CONFIG
READY	I/O	In non-JTAG configuration mode, 1'b1: The device can be programmed and configured. 1'b0: Programming configuration is prohibited
DONE	I/O	1'b1: Successfully programmed and configured. 1'b0: Programming and configuration incomplete.
MODE[2:0]	I, Internal weak pull-up	MODE select (sampled on rising edge of READY)
MCLK	O	SPI output clock
MCS_N	O	SPI chip select, active low
MO	O	SPI output data to Slave
MI	I	SPI Input Data from Slave
FASTRD_N	I	Sampled on rising edge of READY 1'b1: Read SPI mode (SPI instruction:0x03) 1'b0: Fast Read SPI mode (SPI instruction:0x0B)

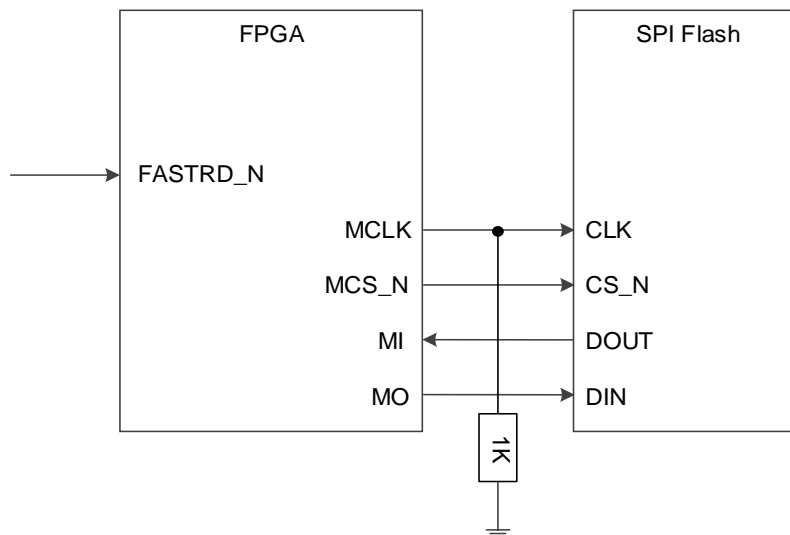
**Note!**

- The MSPI configuration mode clock frequency has a tolerance of  $\pm 10\%$ (Arora family) or  $\pm 5\%$ (LittleBee family).
- The MSPI configuration mode clock frequency should not be greater than 66.6MHz.
- When the clock frequency is greater than 30MHz and less than 66.6MHz, you need to use the high-speed access mode of the Flash and externally pull down the FASTRD\_N pin. After pulling down the FASTRD\_N pin, the clock frequency needs to be greater than 5MHz.
- Leave the FASTRD\_N pin floating if the clock frequency is less than 30MHz.

## 7.5.2 Connection Diagram for MSPI Configuration Mode

The MSPI external Flash interface is shown in Figure 7-50.

Figure 7-50 Connection Diagram for MSPI Configuration Mode

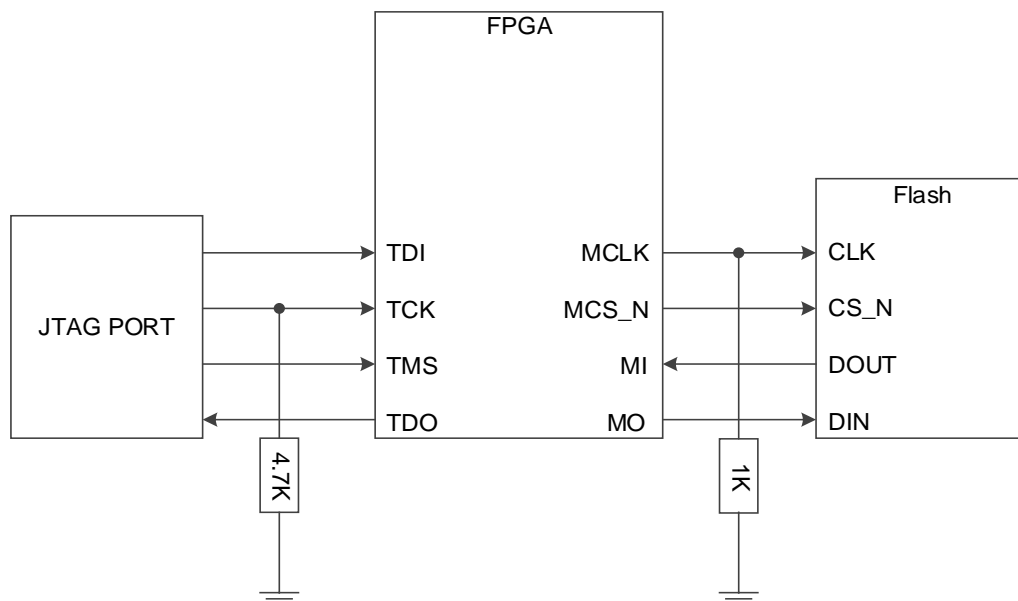


### Note!

The figure above shows the minimum system diagram for the MSPI MODE. The value of the MSPI MODE is "010" (GW1N(R)) and "000" (GW2A(R)). The other fixed pins are shown in Figure 6-1. The FASTRD\_N pin can remain floating in MSPI mode if the clock frequency is less than 30 MHz.

External Flash programming via the FPGA using JTAG is shown in Figure 7-51. The connection diagram for programming external Flash via the SSPI interface is shown in Figure 7-42.

Figure 7-51 Connection Diagram of JTAG Programming External Flash



### Note!

The figure above shows the minimum system diagram of programming external Flash via JTAG. The connection for the other fixed pins is shown in Figure 7-1.

### 7.5.3 MSPI Mode Configuration Attempts

Gowin FPGA usually support just one automatic MSPI configuration attempt following power-up.

The GW1N(R)-9, GW2A(R)-18, and GW1NS series products are improved: If MSPI fails to configure following power up, the device above will automatically attempt to reconfigure.

- The GW2A(R)-18 series FPGA support a total of two configuration attempts.
- GW1N (R)-9 and GW1NS FPGA support a total of three configuration attempts.
- Factors that can lead to a failed configuration include invalid device ID, CRC failure or a false instruction.

The user can specify an alternative SPI Flash start address for the next retry attempt when a bitstream configuration fails. This feature reduces the risk of a configuration failure and can also be used to load a fallback or golden image if a configuration failure occurs.

**Note!**

- If there is an ID Code error or a bitstream header instruction error, it will not boot from the specified SPI Flash address.

The alternative SPI Flash start address is specified using the GOWIN EDA tool Bitstream option when running Design Place & Route (see Multiboot for more details).

### 7.5.4 MULTI BOOT

MULTIBOOT refers to the FPGA reading bitstream data from different addresses in the external Flash memory. MULTIBOOT is supported by all FPGA devices that supports MSPI mode.

The default Flash start address following FPGA power-up is 0x0000 and this address is always used to load the initial bitstream.

The Gowin Programmer software supports the ability to write multiple bitstreams to external Flash at different start addresses without erasing the Flash contents.

When generating a bitstream using the GOWIN EDA tools, the user can specify the SPI Flash start address of the next bitstream to be loaded. I.e the current bitstream header includes a jump address to the next bitstream location in the Flash memory.

At power-on the FPGA will automatically attempt to boot from Flash Addr 0x0000.

If this first boot attempt fails and the FPGA device supports more than one configuration attempt, then the next boot attempt will use the bitstream image specified by the SPI Flash Jump Address provided in the current bitstream header. If the next boot attempt also fails, then this process is repeated until the total number of configuration attempts supported by the FPGA device is exhausted.

Once the FPGA is powered-up, the RECONFIG\_N input can also be pulsed-low to prompt the jump to the next bitstream, where the SPI Flash Jump Address is again provided in the current bitstream header. Note,

there is no limit placed on the number of RECONFIG\_N events.

### Using MULTI BOOT with Failsafe Golden Images

To support remote ‘infield’ bitstream updates, Multiboot can include a failsafe Golden Image. We recommend this Golden (fallback) Image is always stored as the last bitstream in external Flash. In the example below, if Working Images 0x0 or 0x1 are corrupted, we can pulse RECONFIG\_N low to load the next bitstream, using the SPI Flash Address in the current image header as a jump address. This Jump Address could either be the start address of the next Working Image in Flash, or the start address of the Golden (fallback) Image in Flash.

Additionally, if all Working images are erased, then the FPGA will continue reading Flash Addresses until the Golden Image is reached.

In the unlikely event all the Flash images are corrupted, the SPI Flash will need to be reprogrammed via the JTAG/SSPI interface.

Figure 7-52 Example of Bitstream Image Distribution in Flash Memory



For example, **Working Image 0x0** resides at the default 0x0000 power-on address.

**Working Image 0x0** includes a SPI Flash Jump Address to **Working Image 0x1**.

**Working Image 0x1** includes a SPI Flash Jump Address to **Golden Image 0x2**.

Following Power-On, **Working Image 0x0** will automatically be loaded from 0x0000.

If the 1<sup>st</sup> **Working Image 0x0** load fails and the FPGA supports more than one configuration load attempt, the FPGA will then try to load next **Working Image 0x1**.

If the 2<sup>nd</sup> **Working Image 0x1** load fails and the FPGA supports more than two configuration load attempts, the FPGA will then try to load

**Golden Image 0x2.**

If the 1<sup>st</sup> **Working Image 0x0** load fails and the FPGA does not support more than one configuration load attempt, RECONFIG\_N can be pulsed-low to load next **Working Image 0x1**.

If the 2<sup>nd</sup> **Working Image 0x1** load fails and the FPGA does not support more than two configuration load attempts, RECONFIG\_N can be pulsed-low to load **Golden Image 0x2**.

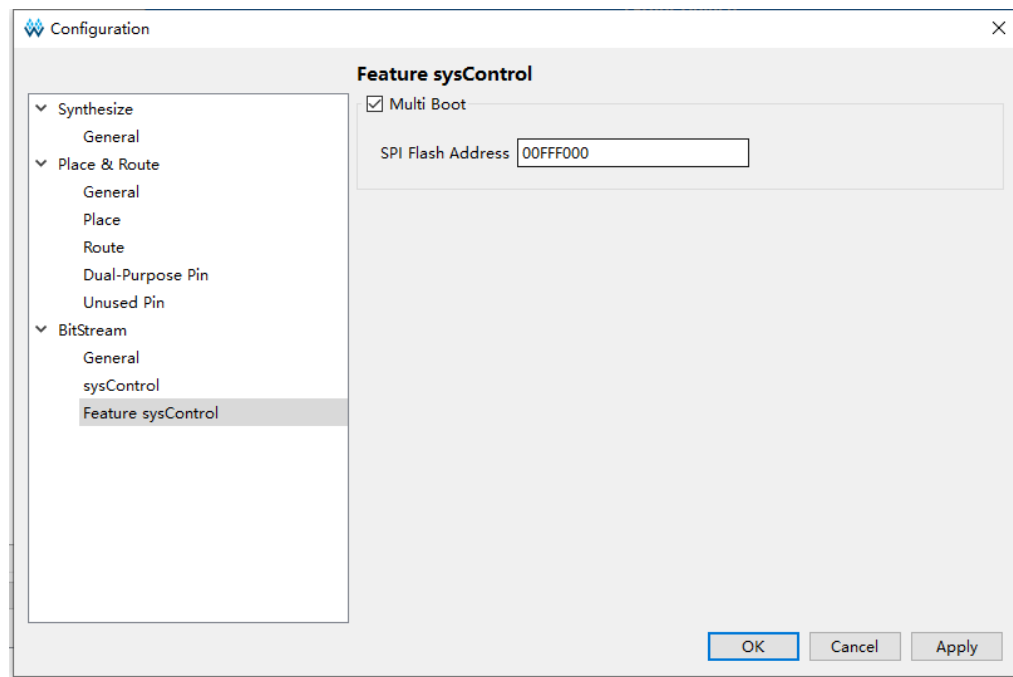
**SPI Flash Start Address**

When generating a bitstream using the GOWIN EDA tools, the user can specify the SPI Flash start address of the next bitstream to be loaded.

Using the GOWIN EDA software, open the "Bitstream" option dialog box.

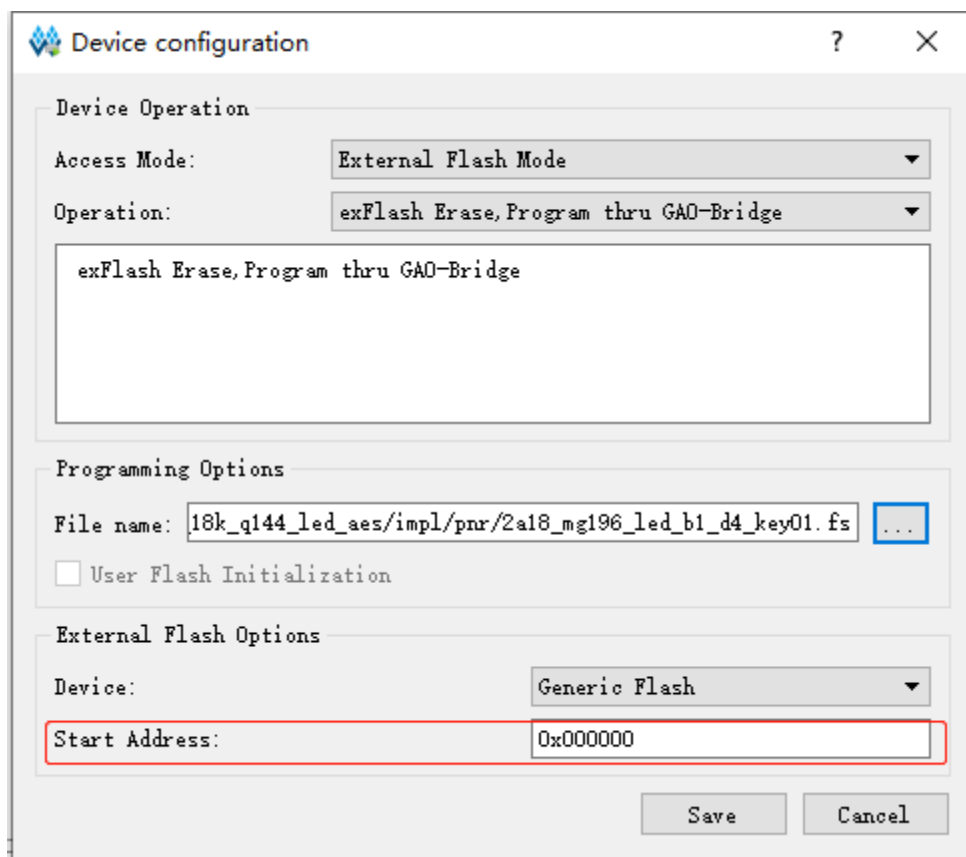
Input the start address for the next Bitstream in the text box following "SPI Flash Address", as shown in Figure 7-53.

**Figure 7-53 Input the Start address for the Next Bitstream**

**SPI Flash Programming**

The Gowin Programmer software supports the ability to write multiple bitstreams to external Flash at different start addresses without erasing the Flash contents.

1. Using GOWIN Programmer Software, select "External Flash Mode", then enter the Bitstream start address in Flash as shown in Figure 7-54.

**Figure 7-54 Set the Programming Address for the External Flash**

2. Click "Save" to complete the setting of Bitstream start address and programming address.

**Note!**

- The Flash start address is reset at power-up.
- When saving multiple images to Flash, you need to calculate the size of the bitstream data to ensure the next start address does not over-write the previous bitstream data.
- The lower 12 bits of the SPI Flash start address are reserved so only address bits ADDR [23:12] can be configured by the user.

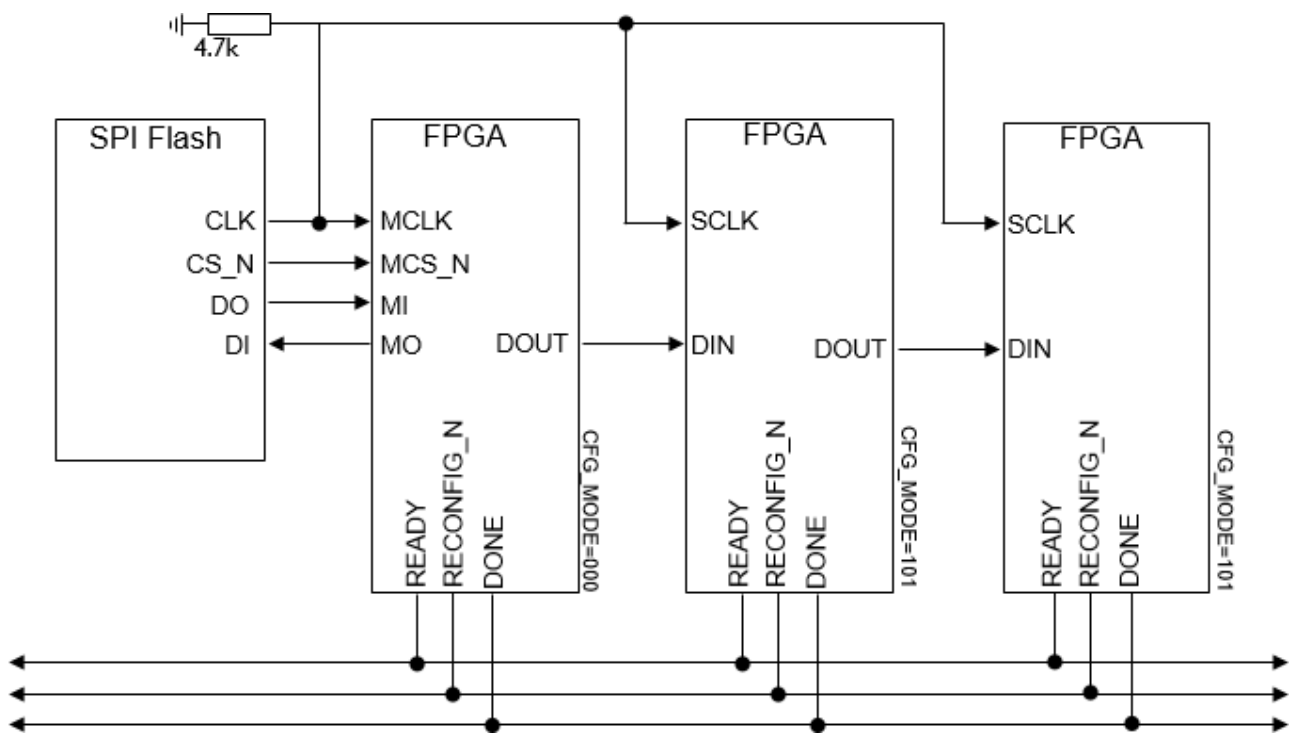
**SPI Flash Programming Multiple FPGA**

Gowin supports configuring multiple FPGAs using a single Flash memory. The 1<sup>st</sup> FPGA is connected directly to the SPI Flash using MSPI mode, while the downstream FPGA devices are configured using SERIAL mode. The Multi FPGA SPI Flash connection diagram is shown in Figure 7-55.

**Notes!**

- For devices that need to forward data, the Wake Up Mode value should be set to 1. Wake Up Mode is usually used in a daisy-chain environment, such as when using an external Flash to configure multiple FPGAs. For more information on Wake Up Mode, please refer to [SUG100, Gowin Software User Guide](#).
- Before configuring, set the MODE value of the 1<sup>st</sup> FPGA to be MSPI and the mode value of the downstream FPGAs to be SERIAL.
- Gowin FPGA products do not support the configuration of one FPGA using multiple Flash devices.

Figure 7-55 Connection Diagram for Configuring Multiple FPGAs via Single Flash



### 7.5.5 MSPI Configuration Timing

MSPI Download Timing is as shown in Figure 7-56.

Figure 7-56 MSPI Download Timing

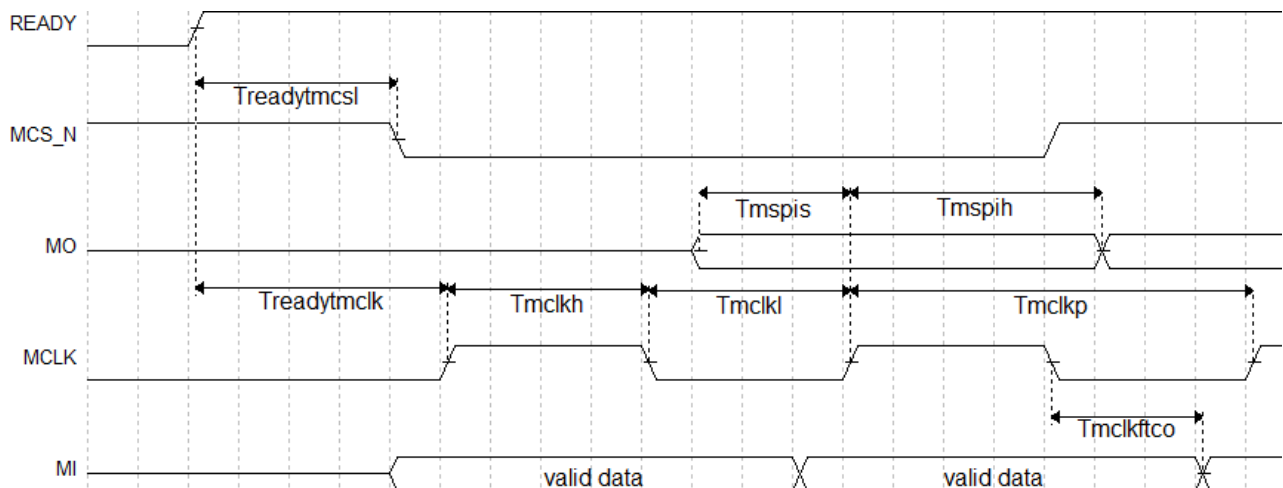


Figure 7-16 shows the timing parameters.

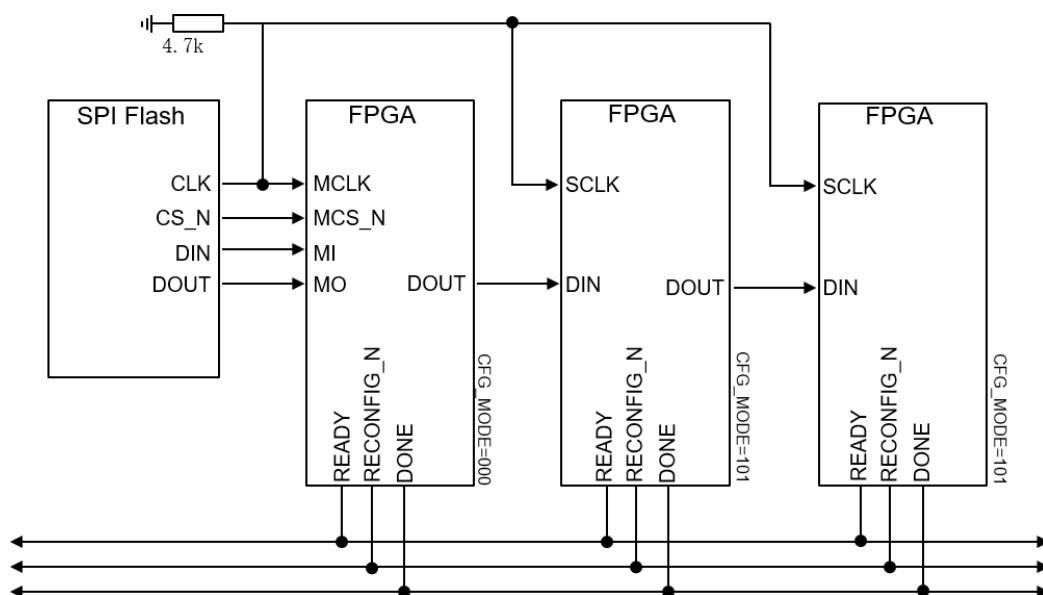
**Table 7-19 MSPI Configuration Timing Parameters**

Name	Description	Min.	Max.
$T_{mclkp}$	MCLK clock period	15ns	-
$T_{mclkh}$	MCLK clock high time	7.5ns	-
$T_{mclkl}$	MCLK clock low time	7.5ns	-
$T_{mspis}$	MSPI PORT setup time	5ns	-
$T_{mspih}$	MSPI PORT hold time	1ns	-
$T_{mclkftco}$	Time from MCLK falling edge to output	-	10ns
$T_{readytmcs}$	Time from READY rising edge to MCS_N low	100ns	200ns
$T_{readytmclk}$	Time from READY rising edge to first MCLK edge	2.8 $\mu$ s	4.4 $\mu$ s

Other than the power requirements, the following conditions need to be met to use the MSPI configuration mode:

- MSPI port enable  
RECONFIG\_N is not set as a GPIO during the first configuration after power up or the previous programming.
- Initiate new configuration  
Power cycle (i.e. power up again) or trigger the RECONFIG\_N pin with a low level pulse.

**Figure 7-57 Multiple FPGA Connection Diagram in MSPI Configuration Mode**



## 7.6 DUAL BOOT Configuration Mode (Supported by LittleBee® Family Only)

The DUAL BOOT mode is a configuration mode supported by the nonvolatile LittleBee® Family of FPGA products. In DUAL BOOT mode, FPGA first reads bitstream data from external Flash to complete configuration.

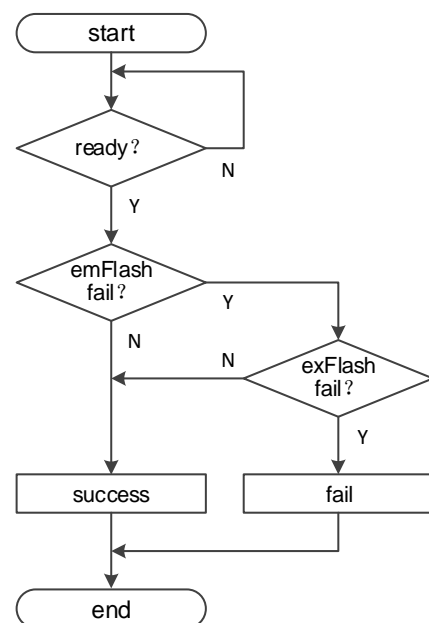
### Note!

In DUAL BOOT mode, when the external Flash is empty or non-existent, FPGA will try to read data from the built-in Flash.

The specific MODE value needs to be selected for the DUAL BOOT MODE. No external connection is required for the built-in Flash. The connection diagram for reading from external Flash is the same as that of the MSPI mode. Please refer to Figure 7-50. In Dual BOOT mode, users can select where to save the configuration data required.

The Dual Boot mode configuration flow is shown in Figure 7-58.

Figure 7-58 Dual Boot Flow Chart



### Note!

When the MODE value is set to "110", the FPGA first attempts to configure from the external Flash.

GW1N(R)-9 and GW1NS series products support four times configuration in all DUAL BOOT modes.

- Start from the preferred storage path and attempt three times; if all attempts fail, start from the other storage path. The embedded Flash can only be started at "0" address.
- When the MODE value is "110", different startup addresses can be selected for the three attempts to start from external Flash. The startup address needs to be written to the bitstream through Gowin Software in advance. If the configuration fails three times, the devices attempt to start from the built-in Flash.

- The GW1NS series of FPGA products support multiple restarts after failures, but the start address cannot be modified.

**Note!**

The lower 12 bits of an SPI Flash startup address is invalid and the address space of ADDR [23:12] can be set by users.

GW1N (R)-4 devices do not currently support automatic DUALBOOT configuration. Gowin provides users with DUAL BOOT configuration solution for these two devices. Please refer to [TN101-1.0, GW1N-4 FPGA Download DUAL BOOT Program](#) for more details.

## 7.7 CPU Configuration Mode

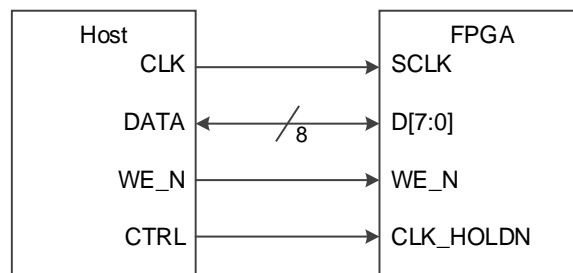
In CPU mode, the Host configures Gowin FPGA products through the 8-bit data bus interface. CPU mode pins are shown in Table 7-20.

Table 7-20 CPU Mode Pins

Pin Name	I/O	Description
RECONFIG_N	I, internal weak pull-up	Low level pulse: Start GowinCONFIG
READY	I/O	High-level pulse: The device can be programmed and configured. Low level: Programming configuration for device is prohibited
DONE	I/O	High-level: Successfully programmed and configured. Low-level: Programming and configuration uncompleted or failed.
MODE[2:0]	I, internal weak pull-up	Configuration mode selection, READY rising edge sampling
SCLK	I	Input clock
CLKHOLD_N	I, internal weak pull-up	Chip select signal in CPU mode, active low. This signal MUST be low to configure the FPGA in CPU mode.
WE_N	I	Read-write enable 0: Write 1: Read
D[7:0]	I/O	Data I/O port: Used as input pin in CPU mode, and used as output pin after configuration for verification

The connection diagram for the CPU mode is shown in Figure 7-59.

Figure 7-59 Connection Diagram for CPU Mode



**Note!**

The figure above shows the minimum system diagram of the CPU MODE. The MODE value is set to "111". The connections for the other fixed pins are shown in Figure 7-1.

Other than the power requirements, the following conditions need to be met to use the CPU configuration mode:

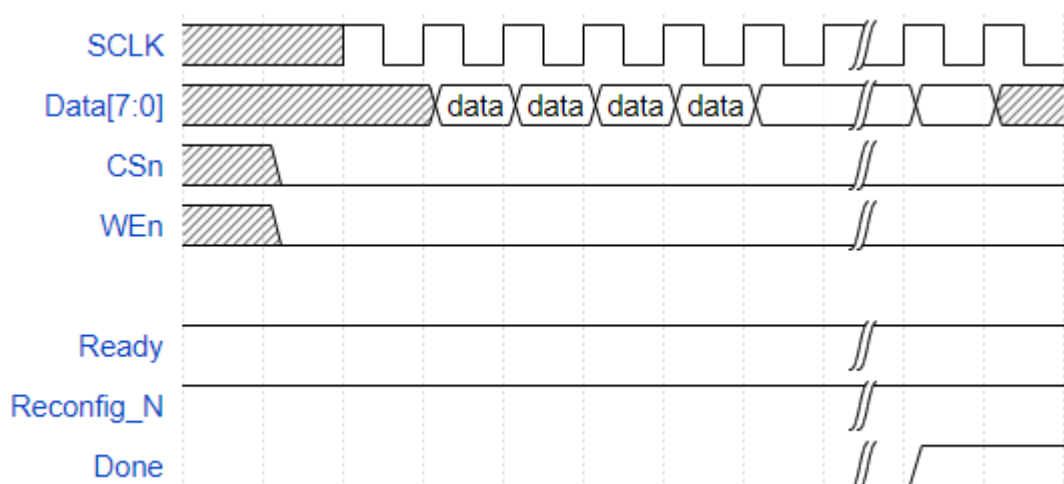
- CPU port enable  
RECONFIG\_N is not set as a GPIO during the first configuration after power up or the previous programming.
- Initiate new configuration  
Power cycle (i.e. power up again) or trigger the RECONFIG\_N pin with a low level pulse.

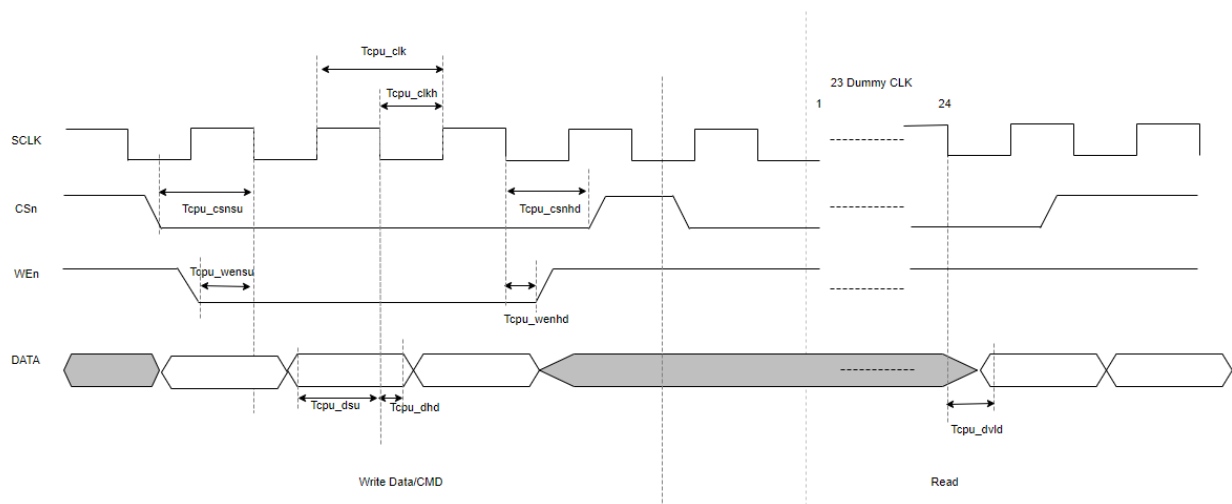
## 7.7.1 Configuration Timing

Before configuration, make sure that MODE[2: 0]=111, and DONE will be pulled up after configuration. If DONE or READY is pulled down, the configuration fails.

In the configuration process, data bus D[7:0] is the MSB mode, and the FPGA reads the data at the SCLK falling edge.

Figure 7-60 CPU Mode Configuration diagram



**Figure 7-61 CPU Mode Configuration Timing****Table 7-21 CPU Configuration Timing Parameters**

Name	Description	Min.	Max.	Units
Tcpu_clk	CPU input clock period	40	—	ns
Tcpu_csnsu	CLKHOLD_N(CSn) setup time to SCLK falling	8	—	ns
Tcpu_csnhd	CLKHOLD_N(CSn) hold time from SCLK falling	0	—	ns
Tcpu_wensu	WE_N setup time to SCLK falling	8	—	ns
Tcpu_wenhd	WE_N hold time from SCLK falling	0	—	ns
Tcpu_dsu	Write data input setup time to SCLK falling	10	—	ns
Tcpu_dhd	Write data input hold time from SCLK falling	0	—	ns
Tcpu_dvld	SCLK falling to read data output valid	—	10	ns
Tcpu_clkh	CPU input clock high duration	(clock cycle ) *45%	(clock cycle) *55%	—

## 7.8 SERIAL Configuration Mode

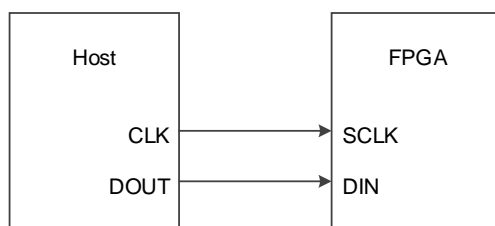
In SERIAL mode, Host configures Gowin FPGA products via serial interface. SERIAL is one of the configuration modes that use the least number of pins. The SERIAL mode can only write bitstream data to FPGA and cannot readback data from FPGA devices; as such, the SERIAL mode cannot read information on the ID CODE and USER CODE and status register. A definition of the pins employed in the SERIAL mode is provided in Table 7-22.

**Table 7-22 Pin Definition in SERIAL Configuration Mode**

Pin Name	I/O	Description
RECONFIG_N	I, internal weak pull-up	Low level pulse: Start GowinCONFIG
READY	I/O	High-level pulse: The device can be programmed and configured. Low level: Programming configuration for device is prohibited
DONE	I/O	High-level: Successfully programmed and configured. Low-level: Programming and configuration uncompleted or failed.
MODE[2:0]	I, internal weak pull-up	Configuration mode selection, READY rising edge sampling
SCLK	I	Input clock
DIN	I, internal weak pull-up	Input data
DOUT	O	Output data, only used in SERIAL configuration mode when FPGA cascading.

The connection diagram for the SERIAL mode is shown in Figure 7-62.

**Figure 7-62 Connection Diagram for SERIAL Mode**



**Note!**

The figure above shows the minimum system diagram of the SERIAL MODE. The MODE value is set to "101". The connection for the other fixed pins is shown in Figure 7-1.

### SERIAL Configuration Timing

See Figure 7-63 for the timing of SERIAL mode.

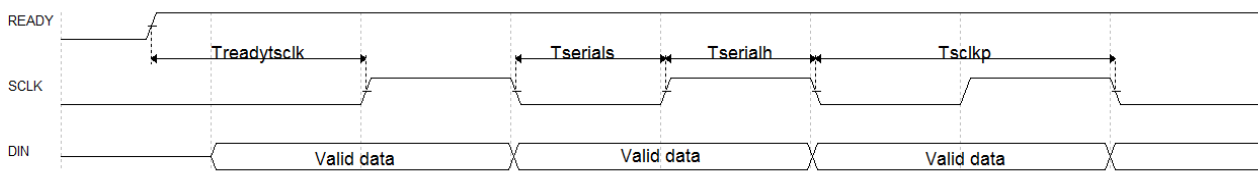
**Figure 7-63 SERIAL Configuration Timing**

Table 7-23 shows the timing parameters.

**Table 7-23 SERIAL Configuration Timing Parameters**

Name	Description	Min.	Max.
$T_{scclkp}$	SCLK clock period	15ns	-
$T_{serials}$	SERIAL PORT setup time	2ns	-
$T_{serialh}$	SERIAL PORT hold time	0ns	-
$T_{readytscclk}$	Time from READY rising edge to first SCLK edge	TBD	-

Other than the power requirements, the following conditions need to be met to use the SERIAL configuration mode:

- SERIAL port enable  
RECONFIG\_N is not set as a GPIO during the first configuration after power up or the previous programming.
- Initiate new configuration  
Power cycle (i.e. power up again) or trigger the RECONFIG\_N pin with a low level pulse.

## 7.9 I<sup>2</sup>C Configuration Mode

### Note!

- Autoboot is automatically enabled in I<sup>2</sup>C mode. In I<sup>2</sup>C mode, following power-on the LittleBee devices will attempt to read data from the internal Flash first. The I<sup>2</sup>C SDA line MUST be held inactive (externally pulled-up) during Autoboot, otherwise the device maynot be configured correctly. Also, it is recommended to externally pull up the SCL line at the same time. Note that this note also applies to C version devices of which the SDA and SCL pins are with internal weak pull-up .
- The internal Flash of the C version GW1N-2 and the C version GW1N-1P5 cannot be programmed via dedicated I<sup>2</sup>C, but it can be programmed using the goConfig I2C IP.

In I<sup>2</sup>C Mode, Gowin FPGA products are configured by Host via I<sup>2</sup>C interface. I<sup>2</sup>C Mode is one of the configuration modes that use the least number of pins. In the I<sup>2</sup>C mode, you can only write bitstream data to the FPGA and cannot read back data from the FPGA. Therefore, the I<sup>2</sup>C mode does not support reading the ID CODE, USER CODE, status register or reading back and verifying the data. A definition of the pins employed in the I<sup>2</sup>C mode is provided in Table 7-24.

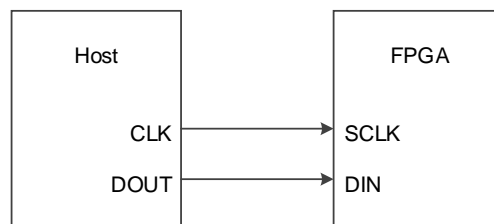
**Table 7-24 Pin Definition in I<sup>2</sup>C Configuration Mode**

Pin Name	I/O	Description
RECONFIG_N	I, internal weak pull-up	Low level pulse: Start GowinCONFIG
READY	I/O	High-level pulse: The device can be programmed and configured. Low level: Programming configuration for device is prohibited
DONE	I/O	High-level: Successfully programmed and configured. Low-level: Programming and configuration uncompleted or failed.
MODE[2:0]	I, internal weak pull-up	Configuration mode selection, READY rising edge sampling
SCL	I <sup>[1]</sup>	Input clock
SDA	I/O <sup>[1]</sup>	Input data or output ACK

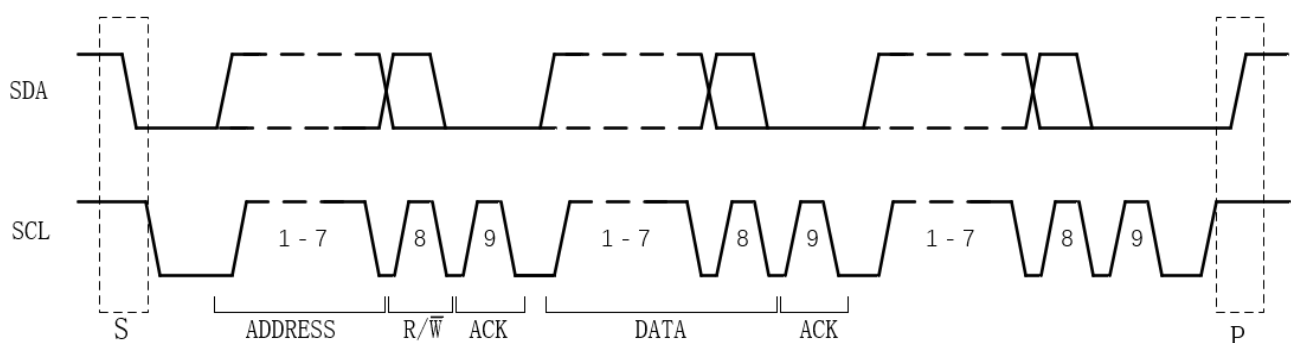
**Note!**

[1] The SCL pin and SDA pin of the C version devices have an internal weak pull-up, but adding a pull-up resistor is still our strongly recommended option.

The connection diagram for the I<sup>2</sup>C mode is shown in Figure 7-64.

**Figure 7-64 Connection Diagram for I<sup>2</sup>C Mode****Note!**

The figure above shows the minimum system diagram of the I<sup>2</sup>C MODE. The MODE value is set to "100". The connection for the other fixed pins is shown in Figure 7-1.

**Figure 7-65 I<sup>2</sup>C Mode Timing**

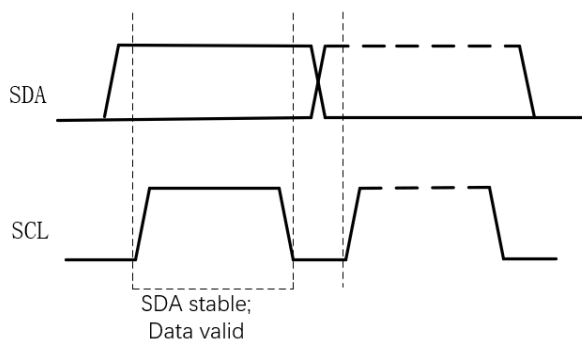
I<sup>2</sup>C is a serial transmission bus, which transmits data according to the protocol shown in the figure above. Under normal status, both SDA and SCL are at high level.

**Table 7-25 I<sup>2</sup>C Configuration Timing Parameters**

Parameter	Description	
S	Startup condition	A HIGH to LOW transition on the SDA line while SCL is HIGH.
P	Stop condition	SDA jumps from low to high while SCL is HIGH.
ADDRESS	Address frame	A unique 7-bit or 10-bit sequence for each slave device that identifies the slave device when the master device is about to communicate with it.
R/W	Read/Write bit	Determines whether the master sends data to the slave (0) or reads data from the slave (1).
ACK	ACK/NACK bit	Each frame in the message is followed by an ACK/NACK bit, and Gowin FPGA returns 0 if correct.
DATA	Data	A data has 8bits, and the most significant bit is sent first.

All DATA on the I2C bus is transmitted in 8-bit bytes. Each byte sent by the transmitter, it releases the DATA line during the clock pulse 9, and the receiver sends back a response signal. The response signal is a valid response bit (ACK bit) if it is low, indicating that the receiver has successfully received the byte. The response signal is a non-acknowledgment bit (NACK) if it is high, which generally indicates that the receiver did not succeed in receiving the byte. The requirement for the ACK feedback is that the receiver pulls the SDA line low during the low level prior to the 9th clock pulse and ensures a stable low level during the high level of the clock. If the receiver is the master, after it receives the last byte, it sends a NACK signal to notify the controlled sender to end the data transmission and releases the SDA line for the master receiver to send a stop signal.

Each bit of data transmitted on the I2C bus has a corresponding clock pulse (or synchronous control), that is, each bit of data is transmitted serially on the SDA bit by bit based on the SCL serial clock. During data transfer, the level on the SDA must remain stable, with the low level being data 0 and the high level being data 1, while the SCL is high. The level on the SDA is allowed to change state only while the SCL is low. Logic 0 has a low voltage level and Logic 1 has a high voltage level, as shown in the figure below.



The list of I<sup>2</sup>C mode supported by Gowin FPGA devices is as shown in the table below.

**Table 7-26 Frequencies and addresses of I<sup>2</sup>C configuration mode**

Mode	Device	Frequency	Address
SRAM	GW1N-2 (IDCode:0x0120681B)	100Khz~1.33Mhz	7'b1010_000

**Note!**

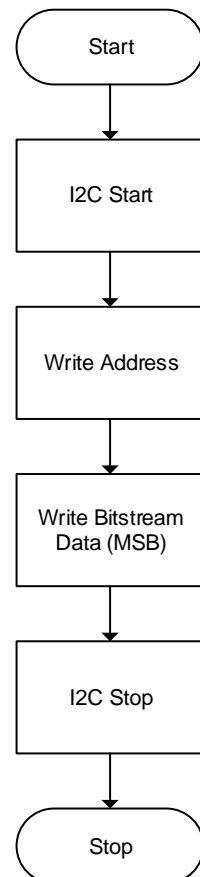
To operate the Flash via I2C, you need to use the goConfig I2C IP.

Other than the power requirements, the following conditions need to be met to use the I<sup>2</sup>C configuration mode:

- I<sup>2</sup>C port enable  
RECONFIG\_N is not set as a GPIO during the first configuration after power up or the previous programming.
- Initiate new configuration  
Power cycle (i.e. power up again) or trigger the RECONFIG\_N pin with a low level pulse.

### 7.9.1 Process of Configuring SRAM of GW1N-2

The data stream file format for configuring SRAM is FS (.fs) or Binary (.bin). Regardless of the file format, the data is sent byte by byte in MSB way.

**Figure 7-66 Process of Configuring SRAM of GW1N-2**

# 8 Safety Precautions

Security is a key factor for users to design FPGA. Combined with GOWINSEMI devices features, Gowin programmer offers a series of safety precautions, which provides a perfect security mechanism for users' bitstream data.

Safety precautions consist of three stages:

- Before configuration, Gowin programmer checks the validity of the bitstream.
- During configuration, GOWINSEMI device verifies the accuracy of the transmission data in real time.
- After configuration, GOWINSEMI device enters the working state, masking any readback requests.

The details of the three stages are as follows:

## Before Configuration

Gowin programmer can be used to configure Gowin FPGA by following the steps outlined below.

1. Connect the device that needs to be configured.
2. Start Gowin programmer to start scanning, and the connected FPGA devices can be identified automatically.
3. Select the bitstream and the configuration mode to configure the device.

During the process outlined above, Gowin programmer will read the connected device ID first, and then compare this with the bitstream ID that users selected. The configuration can only proceed when the two IDs are identical, or the bitstream selected by users will be regarded as illegal data, resulting in configuration failure.

### Note!

GOWINSEMI products have specific IDs that distinguish them from the other series of products. The bitstream generated by Gowin Software contains an ID verification directive, as such, users only need to select the specific device when creating a new project.

### During Configuration

The device reads and verifies the bit stream ID first, and configuration starts if verification passes. To prevent bitstream modifications or possible transmission errors, GOWINSEMI devices adopt CRC to ensure bitstream is written in correctly. The specific process is outlined below.

Following each address segment of the bitstream generated by Gowin Software, a CRC code is added. GOWINSEMI devices generate a CRC code in the process of receiving data and compares them with the check codes received. If a CRC error is detected, any data transmitted following this error will be ignored. The "DONE" indicator will not light up after configuration, and the CRC error message will be displayed on the Gowin programmer interface.

### After Configuration

After configuration, the device bitstream will be loaded to the SRAM or the on-chip Flash according to the configuration mode selected. (On-chip Flash is supported by the LittleBee® Family of FPGA products only.)

- If the data is loaded to the SRAM, Gowin Software sets the security bit automatically in the process of bitstream generation, and no user can read SRAMs.
- If the data is loaded to the on-chip Flash, the Flash will be configured as the AUTO BOOT mode when its programming is complete. Any reading requests will be prohibited.

The AUTO BOOT mode of the LittleBee® Family of FPGA products does not require external connections, so this greatly reduces the risk of data interception and provides the user with higher security. DUAL BOOT provides a selection for users with the option to write the configuration data to off-chip Flash as required.

#### **Note!**

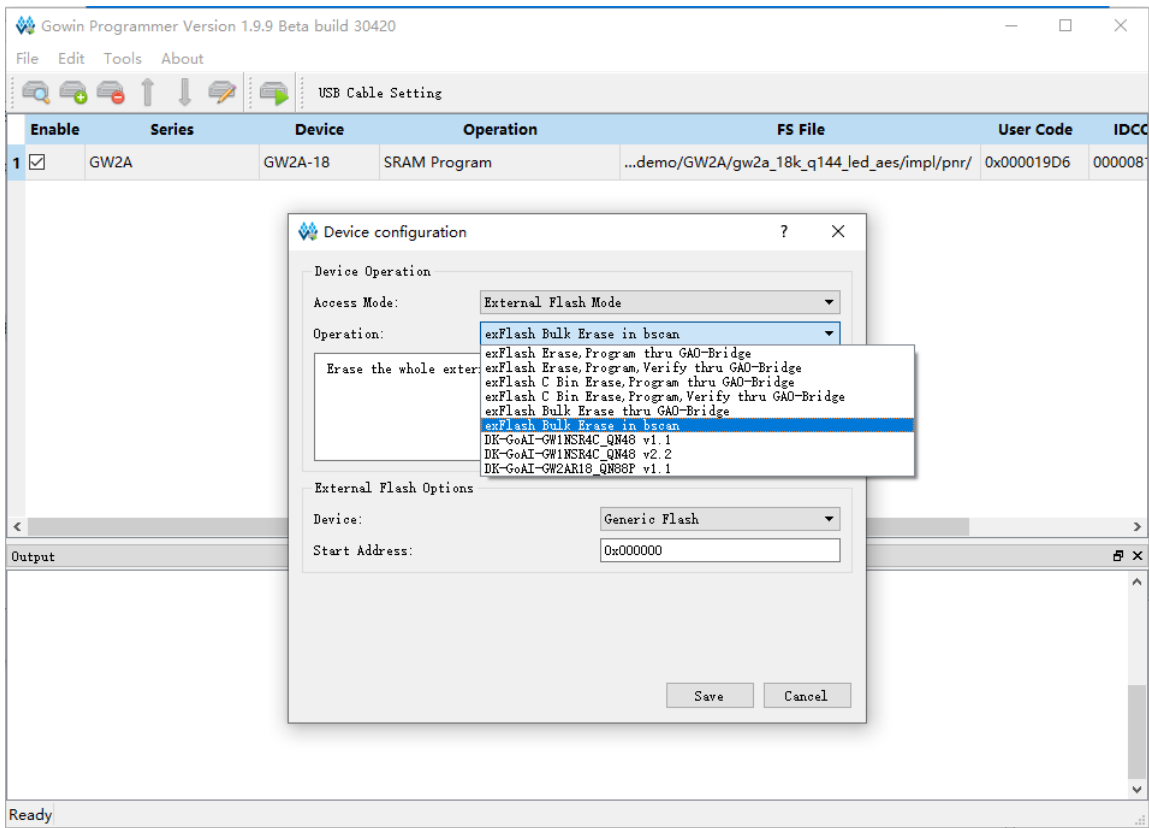
GOWINSEMI takes no responsibility for the security of the external Flash.

# 9 Boundary Scan

The boundary scan operation is an extension of the JTAG configuration mode. The scanning chains contain long chain and short chain. The long chain is mainly combined with BSDL file for device testing. The short chain is mainly used to erase and read and write the external Flash on the FPGA chain.

- To perform a boundary scan, follow the steps outlined below:
1. Connect the FPGA development board to the PC and then power up.
  2. Open Gowin programmer and scan the connected devices.
  3. Double-click in the "Operation" field and select "External Flash Mode" and the related bscan operation, as shown in Figure 9-1.

Figure 9-1 Boundary Scan Operation Schematic Diagram



The boundary scan operation can only be performed on the external Flash of FPGA and cannot be used to program the embedded Flash or SRAM. This operation is irrelevant with the FPGA MODE value, but it is slower than that of the external Flash programming via JTAG.

# 10 SPI Flash Selection

The external SPI Flash device operation commands supported by Gowin FPGA products are shown in Table 10-1.

**Table 10-1 SPI Flash Commands**

Operation	Command
Read	0x03
Fast_Read	0x0B
Page Program	0x02
Sector Erase	0x20
Chip Erase	0xC7
Read Status Code	0x05
Read JEDEC ID	0x9F
Write Enable	0x06
Write Disable	0x04

**Note!**

- At least one of the Flash read commands supported by the Gowin FPGA must be 0x03 or 0x0B. Use the Read command if the clock frequency is no higher than 30 MHz. Use the Fast\_Read command if the clock frequency is higher than 30 MHz. Fast read requires the FASTRD\_N pin to be pulled down, and the clock frequency shall not be higher than 66.6MHz.
- Read(0x03) and Fast\_Read(0x0B) are the only commands supported when the device is in MSPI mode; the other commands are used to program the Flash with Programmer.
- By default the SPI Flash needs to work in Standard SPI protocol.
- Flash products from Winbond, GigaDevice, and ISSI(such as ISSI's SPI-Flash "IS25LP064A-JBLE") are mainly used in our tests. If Flash products from other companies are used, there may be exceptions due to timing differences, even if the above command requirements are met.

